Females' and Males' End-User Debugging Strategies: A Sensemaking Perspective

VALENTINA GRIGOREANU, MARGARET BURNETT
Oregon State University

SUSAN WIEDENBECK
Drexel University

JILL CAO, KYLE RECTOR
Oregon State University

Although there have been decades of research into how professional programmers debug, only recently has work begun to emerge about how end-user programmers attempt to debug their programs. Without understanding how end-user programmers approach debugging, we cannot build tools to adequately support their needs. To help fill this need, this paper reports the results of a qualitative empirical study that investigates in detail female and male end-user programmers' sensemaking about a spreadsheet's correctness. Using our study's data, we derived a sensemaking model for end-user debugging and then categorized participants' activities and verbalizations according to this model. We then used the categorized data to investigate how our participants went about debugging. Among the results are identification of the prevalence of information foraging during end-user debugging, two successful strategies for traversing the sensemaking model, ties to gender differences in the literature, sensemaking sequences leading to debugging progress, and sensemaking sequences tied with troublesome points in the debugging process. The results also reveal new implications for the design of spreadsheet tools to support female and male end-user programmers' sensemaking as they debug.

Categories and Subject Descriptors: D.2.5 **[Software Engineering]**: Testing and Debugging—*debugging aids*; H.1.2 **[Models and Principles]**: User/Machine Systems—*human factors; human information processing; software psychology*; H.4.1 **[Information Systems Applications]**: Office Automation—*spreadsheets*.

General Terms: Human Factors

Additional Key Words and Phrases: End-user programming, end-user software engineering, debugging, debugging strategies, gender differences, gender HCI, sensemaking, spreadsheets

1. INTRODUCTION

Although twenty years ago, the idea of end users creating their own programs was still a revolutionary concept, today, end-user programming has become a widespread phenomenon. In fact, in the U.S., there are now more end-user programmers than professional programmers [Scaffidi et al. 2005]. Today's end-user programmers include anyone who creates artifacts that instruct computers how to perform an upcoming computation. Examples include an accountant creating a budget spreadsheet, a garage mechanic entering rules to sort email, or a teacher authoring educational simulations of science phenomena. The pervasiveness of end-user programming today is in part due to research advances such as graphical techniques for programming, programming by demonstration, and innovative ways of representing programs (described, for example, in [Kelleher and Pausch 2005; Myers et al. 2006; Nardi 1993]), and in part due to the popularity of spreadsheets [Scaffidi et al. 2005].

Along with the ability to create programs comes the need to debug them, and work on end-user debugging is only beginning to become established. The numerous reports of expensive errors in end users' programs, especially

V. Grigoreanu is currently affiliated with the School of Electrical Engineering and Computer Science at Oregon State University in Corvallis, Oregon. Authors' email addresses: i.valentina.g@gmail.com, burnett@eecs.oregonstate.edu, susan.wiedenbeck@cis.drexel.edu, coach@eecs.oregonstate.edu, rectorky@eecs.oregonstate.edu.

spreadsheets (e.g., [Boehm and Basili 2001; Butler 2000; EUSPRIG 2009; Panko 1998; Panko and Orday 2005]), make clear that supporting end users' debugging efforts is important. There has been recent work on tools for end-user debugging to fill this need (e.g., [Abraham and Erwig 2007; Ayalew and Mittermeir 2003; Burnett et al. 2003; Burnett et al. 2004; Ko and Myers 2004; Wagner and Lieberman 2004]), but a key issue that remains largely unanswered is *how* end-user programmers go about debugging. We believe that knowing more about end users' debugging strategies is important to informing the design of better tools to support their debugging.

This paper helps to fill this gap in knowledge by considering end-user debugging from a sensemaking perspective. *Sensemaking* is a term used to describe how people make sense of the information around them, and how they represent and encode that knowledge, so as to answer task-specific questions [Russell et al. 1993]. As we discuss in more detail in Section 2, sensemaking models provide a detailed view of strategies people use when trying to make sense of information they need, in situations in which much of the information available may be irrelevant to the problem at hand. Since such situations are precisely the sort encountered by debuggers—perhaps especially so by end-user debuggers with little training in debugging techniques—we posit that sensemaking is a suitable lens from which to gain new insights into debugging.

To understand how end-user programmers solve debugging problems, it is important to take into account individual differences in problem-solving styles. To gain insights into individual differences, it is often useful to consider identifiable subpopulations, and that is our approach here. One such division that has reported important differences in end-user debugging is gender [Beckwith et al. 2005; Beckwith et al. 2006; Grigoreanu et al. 2006; Grigoreanu et al. 2009; Subrahmaniyan et al. 2008], and as a result, debugging tool improvements have begun to emerge that have been helpful to both males and females [Grigoreanu et al. 2008].

Therefore, in this paper, we investigate the following research question:

How do male and female end-user programmers make sense of spreadsheets' correctness when debugging?

To answer this question, we collected detailed activity logs and think-aloud data from end users debugging a spreadsheet, and used the results to derive from earlier sensemaking research into intelligence analysts [Pirolli and Card 2005] a new model of sensemaking for end-user debuggers. Our model is particularly suited for use with empirical work, because it is expressed solely in terms of the *data* being processed by the user rather than on internal mental activities that do the processing. We use it in this paper to characterize our end-user debugging data in terms of sensemaking sequences, sensemaking subloops, and relationships between sensemaking and debugging. Among the results were the identifications of: the prevalence of information foraging during end-user debugging, two successful ways of traversing the sensemaking model and their ties to literature on gender differences, sensemaking sequences leading to debugging progress, and sensemaking sequences tied with troublesome points in the debugging process. Finally, we discuss how these results can be taken into account to build future tools for end-user debuggers.

2. BACKGROUND AND RELATED WORK

2.1 Debugging by End-User Programmers

There have been decades of research on professional programmers' debugging strategies (see [Romero et al. 2007] for a summary), but the works most related to this paper are those on novice programmers' debugging strategies, end-user programmers' debugging feature usage, and end-user programming. Note that novice programmers are not necessarily the same as end-user programmers. Novice programmers program in order to learn the profession so that they can become professional developers. End-user programmers usually do not aspire to become professional developers; instead, their programming interest is more likely to be in the result of the program rather than the program itself [Nardi 1993]. Such differences in goals can play out in differences in motivation, in degree of intent to achieve software quality, and in the importance placed on understanding the fine points of the program. However, like novices, end-user programmers usually do not have professional programming experience, and therefore research into novice programmers' debugging is pertinent to end-user debugging.

Given that novice programmers program in order to learn, a number of researchers have looked into how novice programmers gain skill. One recent effort in this direction was research into the learning barriers faced by this population [Ko et al. 2004], which reported barriers of six types: design, selection, coordination, use, understanding, and information. Although the discussion of these relate mainly to the context of creating a new program from scratch, the barriers also tie to debugging, since difficulties with understanding a program's behavior can lead the programmer to a debugging mode. In fact, research on novice programmers shows that program comprehension is key to successful debugging (e.g., [Jeffries 1982; Nanja and Cook 1987]). For example, fixing a program with multiple modules can become intractable if the programmer does not understand the dependencies of the modules [Littman et al. 1986]. Also, *how* a programmer goes about comprehending a program matters. For example, reading a program in the order in which it will be executed has been empirically shown to be superior to reading the program from beginning to end like text [Jeffries 1982]. The essence of previous research into novice programmers points to their need for a sound understanding of the high-level structure of the program and the interactions among parts of the program in order to debug or maintain software effectively. The literature on novice programming was summarized in [Kelleher and Pausch 2005; Pane and Myers 1996].

To be precise about understanding strategies, we first introduce the nuances between two related terms employed in this paper: strategems and strategies. A *strategem* is a complex set of thoughts and/or actions, while a *strategy* is a plan which may contain strategems for the entire task [Bates 1990]. For the remainder of this section, we focus on empirical findings and theories about end-user programmers' debugging behaviors and strategems.

Empirical research of end users' debugging has begun to appear in recent years [Abraham and Erwig 2007; Beckwith et al. 2005; Beckwith et al. 2006; Beckwith et al. 2007; Fern et al. 2009; Grigoreanu et al. 2006; Grigoreanu et al. 2008; Grigoreanu et al. 2009; Kissinger et al. 2006; Ko and Myers 2004; Phalgune et al. 2005; Prabhakararao et al. 2003; Rode and Rosson 2003; Subrahmaniyan et al. 2008]. One useful finding relating to how end users make sense of their programs' flaws is Rode and Rosson's empirical work showing how users "debugged their programs into existence" [Rode and Rosson 2003]. That is, they began with an initial solution, then iteratively found flaws with it, and expanded their solution to correct those flaws, which required adding new functionalities at the same time, debugging that new functionality to uncover more flaws, and so on. One difficulty in such debugging efforts has been end users' difficulties judging whether the program is working correctly [Phalgune et al. 2005]. To inform supporting end-users' debugging, studies investigating end users' information needs during debugging revealed that much of what end-user programmers wanted to know during debugging was "why" and "why not" oriented [Ko and Myers 2004], and that they also wanted to know more about strategies for debugging, not just features for doing so [Kissinger et al. 2006].

A reason this paper explicitly considers the gender of participants in our analysis is the recent body of research suggesting that females and males may go about debugging (and other software development tasks) differently [Beckwith et al. 2005; Beckwith et al. 2006; Beckwith et al. 2007; Grigoreanu et al. 2008; Ioannidou et al. 2008; Kelleher et al. 2007; Rosson et al. 2007]. For example, in spreadsheet debugging, females' self-efficacy (confidence about spreadsheet debugging) [Bandura 1986] has been lower than males', as has their willingness to use new debugging features, which in some cases led to lower performance outcomes [Beckwith et al. 2005; Beckwith et al. 2006]. In contrast to the females in these studies, the males' self efficacy was not correlated with their willingness to use the same new features. These studies are consistent with other studies of females' and males' technical and quantitative tasks revealing females to have lower self-efficacy than males in such tasks [Torkzadah and Koufteros 1994; Busch 1995; Gallagher et al. 2000; Hartzel 2003]. Research also reports that females tend to be more risk-averse than males [Byrnes et al. 1999; Finucane 2000; Powell and Ansic 1997]. The attributes of risk averseness and low self-efficacy are related, and may snowball. For example, risk-averse females who try out new debugging features and are not immediately successful may experience further reduced self-efficacy as a result, thereby enhancing the perception of risk in adopting the features.

Pertinent to end users' sensemaking about program bugs, Meyers-Levy's Selectivity Hypothesis describes two strategies in how people process information [Meyers-Levy 1989]. According to the Selectivity Hypothesis, males prefer to use a heuristic (or selective) approach that involves striving for efficiency by following contextually salient cues, whereas females process information comprehensively, seeking a more complete understanding of the problem. Empirical work supports this theory [Meyers-Levy 1989; O'Donnell and Johnson 2001], some of which has taken place in the context of an end-user programming tasks. In a study of spreadsheet auditing, female auditors were statistically more efficient (completed the task in less time and used fewer information items) than males in a complex analytical procedures task through use of comprehensive information processing, whereas males were statistically more efficient (used fewer information items) than females in a simple task through use of selective information processing [O'Donnell and Johnson 2001]. Research into female and male effective end-user debugging strategems also seem related to their preferred information processing styles: females' (but not males') success has been tied to the use of code inspection, whereas males' (but not females') was tied to dataflow [Grigoreanu et al. 2009; Subrahmaniyan et al. 2008]. In both of these studies, females were observed to use the elaborative information processing with code inspection to examine formulas broadly and in detail to get the big picture of the spreadsheet. Males' use of dataflow, on the other hand, was to follow a particular formula's dependencies, in essence being selective about the information being pursued by going for depth early, but at the expense of a comprehensive understanding of the spreadsheet.

2.2 Sensemaking

Dervin began developing human-centric models in 1972, creating the Sensemaking methodology for the design of human communication systems. This methodology was grounded in her model of how a person makes sense of his or her situation, referred to as the *sensemaking triangle.* The most complete presentation of Dervin's early work on the sensemaking triangle model is [Dervin 1984], and a more modern overview of her sensemaking work is [Dervin et al. 2003]. According to Dervin's triangle model, an individual trying to make sense of a complex situation steps through a space-time context. Beginning with the current situation (what he or she knows now), the individual then moves through the space to recognizing gaps in understanding (questions, confusions, muddles) that must be "bridged" via resources (ideas, cognitions, beliefs, intuitions), leading to analysis and outcomes (effects, consequences, hindrances, impacts). Although Dervin's work mostly appeared in Communications literature, her goals align with those of Human-Computer Interaction (HCI) research that aims to design and implement human-centric communication systems [Dervin et al. 2003].

A series of sensemaking models later began to appear in Computer Science literature, particularly HCI. These new models focused primarily on Dervin's "bridge" aspect of sensemaking. An early effort in this regard, before the term "sensemaking" had been adopted by the HCI community, was Shrager and Klahr's experiment on instructionless learning [Shrager and Klahr 1986]. Non-technical participants were given a programmable toy and told to figure it out, without any manual or help. The study revealed that participants went through an orientation phase followed by a hypothesis phase: after direct attempts to control the device failed, participants systematically formulated hypotheses about the device by observation and tested them by interacting with the device. If hypotheses were not confirmed, the participants refined and tested them iteratively. The authors used these results to form a theory of how users make sense of a program based on its outputs, with a focus on participants' partial schemas developed through observation and hypothesis testing.

One of the earliest appearances of the term "sensemaking" in HCI was Russell et al.'s work on a model of the cost structure of sensemaking [Russell et al. 1993] known as the Learning Loop Complex. In this work, the authors observed how Xerox technicians made sense of laser printers: they entered a learning loop to seek suitable representations of the problem, then instantiated those representations, then shifted the representations when faced with residue (e.g., ill-fitting data, missing data, unused representations), and finally consumed the encodons (i.e., instantiated schemas) and representations they created [Russell et al. 1993].

Since that time, other versions of sensemaking models have emerged for different domains, of which Stefik et al. provide a good overview [Stefik et al. 2002]. Most of these models depict roughly the same sort of process, each providing a progression by which existing information or knowledge is turned into new knowledge useful to a target task. Even so, there remains controversy over what exactly sensemaking is and how sensemaking research should be performed. These differences in perspective are summarized in [Leedom 2001; Klein et al. 2006]. As these summaries point out, the psychological perspective focuses on creating a mental model, and takes into account elements that are not sensemaking per se, but may contribute to sensemaking, including creativity, curiosity comprehension, and situation awareness. The naturalistic decision-making perspective on sensemaking is empirically based and keeps expert analysts in the center of the sensemaking process, but uses decision aids as needed to improve the process. The human-centered computing perspective critiques intelligent systems intended to automatically solve the problem of sensemaking. For example, in intelligent systems, fused data from multiple sources reduce information overload, but hide the data from the analyst, which negates the analysts' expertise and prevents the development of their own interpretations [Leedom 2001; Klein et al. 2006].

Of particular interest to this paper is the relatively recent Pirolli/Card Sensemaking Model for Analysts [Pirolli and Card 2005]. As with other sensemaking research, Pirolli and Card pointed out that sensemaking is not based on direct insights or retrieving a final answer from memory, but rather a process that involves planning, evaluating, and reasoning about alternative future steps [Pirolli and Card 2005]. Their sensemaking model (Figure 1) can be viewed as a more detailed version of Russell et al,'s Learning Loop Complex. Like the Russell et al. model, the Pirolli/Card model focuses on how users' representations of data change during sensemaking. Pirolli and Card derived this model through cognitive task analysis and think-aloud protocol analysis of intelligence analysts' sensemaking data. The derived overall process is organized into two major loops of activities: first, a foraging loop for searching, filtering, reading, extracting information, and second, a sensemaking loop for organizing the relevant information into a large structure that leads to knowledge products. (These loops are coupled in Russell et al.'s works too [Russell et al. 1993; Furnas and Russell 2005].) The Pirolli/Card model can be viewed as a model with a low-level focus on Dervin's "bridge" component and Russell et al.'s learning loop complex," and this low-level focus made it a good fit for our interest in investigating end-user debugging in a fine-grained way. We will thus return to the Pirolli/Card model in Section 5.

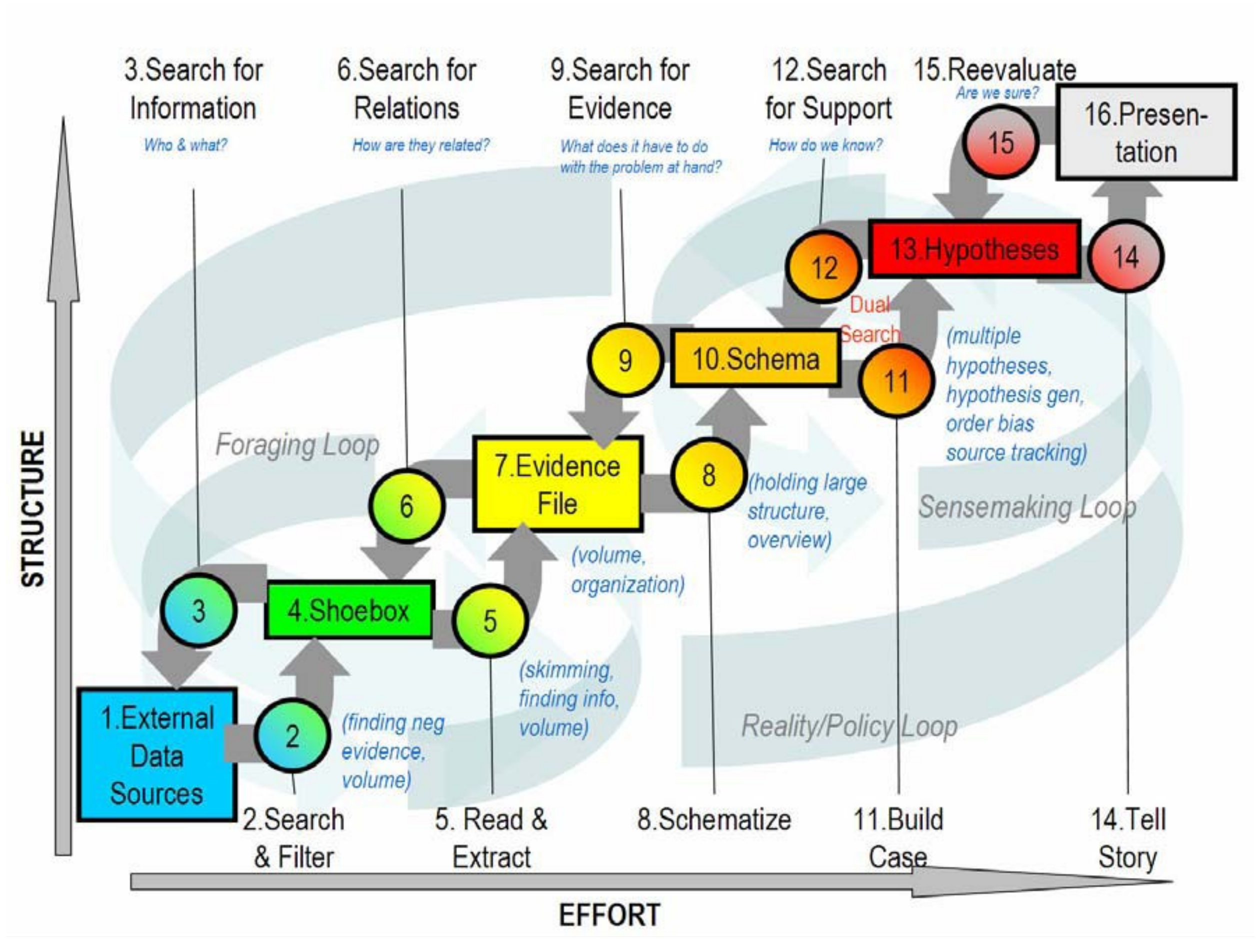


Figure 1. Pirolli and Card's sensemaking model for intelligence analysts. The rectangles represent how the data representation changes at each step, while the arrows represent the process flow [Pirolli and Card 2005].

3. THE SENSEMAKING STUDY

3.1 Procedure

The study, conducted one participant at a time, used the think-aloud method. Each participant first read and signed the informed consent paperwork. We then conducted a pre-session interview about their spreadsheet background, which covered whether the participant had previously used spreadsheets for school, work, or personal reasons, and about the types of formulas he or she had written (Table 1). We then gave the participant a brief tutorial and warm-up exercise, described in Section 3.4, to ensure familiarity with Excel's audit features and with verbalizing during the task.

Each participant was asked to "make sure the spreadsheet is correct and, if [you] find any errors, fix them." We also provided the paper handout, shown in Figure 2, to give an overview of the way the spreadsheet was supposed to work. The participants had 45 minutes to debug the spreadsheet. The data we captured during the sessions were video showing their facial expressions, audio of their think-aloud verbalizations, synchronized screen recordings of the entire session (including pre-session background interviews and the task itself), and their final Excel spreadsheets.

3.2 Participants

Ten participants (five men and five women) participated in our study. To take part in our study, participants were required to have experience with Excel spreadsheet formulas. Background in computer science was not allowed beyond the rudimentary requirements of their majors. Participants received a $20 gratuity for their participation.

Table 1: Study participants' spreadsheet background.

Participant	Major	Spreadsheet Experience	Computer Science Background
P05200830 (Male)	Biochemistry / Biophysics (undergraduate)	School: chemistry labs.	No programming experience.
		Formula experience: standard deviations, quadratic formulas, and basic arithmetic.	
P05211130 (Female)	Business Administration (undergraduate)	School, work, personal: checking paperwork (bank statements, personal records).	An undergraduate business class on Business Application Development.
		Formula experience: SUM, IF, SUB, other basic functions.	
P05211600 (Male)	Animal Science (graduate)	School, work, personal: in classes, manages his own and his dorms' finances.	An introductory computing class with some HTML.
		Formula experience: basic arithmetic, statistics, and calculus.	
P05220830 (Male)	Mechanical Engineering (undergraduate)	School, work, personal: created a spreadsheet for timing races and others for running a club.	A Q-BASIC class in high school and is familiar with MATLAB.
		Formula experience: basic formulas, VLOOKUP, embedded IF, etc.	
P05221230 (Male)	Mechanical Engineering (undergraduate)	School and work: works as an accountant.	No programming experience.
		Formula experience: AVERAGE, MIN, MAX, COUNT, COUNTA.	
P05270830 (Female)	Rangeland Ecology and Management (graduate)	School, work, personal: was an accountant six years, and now uses it for her research and labs.	No programming experience (wrote a few macros years ago, but did not think she remembered how).
		Formula experience: basic Excel formulas, IF, statistics formulas.	
P05281130 (Female)	Pharmacy (undergraduate)	School: spreadsheets for labs and graphs/charts for reports.	An introductory CS class.
		Formula experience: SUM, AVERAGE, MAX, MIN, and basic arithmetic.	
P05290830 (Female)	Animal Science (undergraduate)	School, work: was a club treasurer and calculated interest rates for classes.	No programming experience.
		Formula experience: basic arithmetic, statistical formulas, and also financial formulas (N, PMT, FV, PV, I).	

We discarded the data of a participant whose verbalizations were inaudible and the data of a participant whose Excel experience was so much lower than he had claimed during recruitment that he was unable to proceed with the debugging task. The remaining eight participants were undergraduate and graduate students at Oregon State University majoring in animal science, biochemistry, business administration, mechanical engineering, pharmacy, and rangeland ecology/management. Table 1 details participant backgrounds.

3.3 Task and Materials

The spreadsheet the participants debugged was fairly complex. It consisted of the worksheet thumbnailed in Figure 2 and a small second sheet that produced frequency statistics and a chart of the grades calculated in the main sheet. We obtained the spreadsheet from the EUSES Spreadsheet Corpus of real-world spreadsheets [Fisher II and Rothermel 2005], most of which were obtained from the web. We chose the spreadsheet for the following reasons. First, it was complicated enough to ensure that we would obtain a reasonable amount of sensemaking data. Second, its domain (grading) was familiar to most participants, helping to avoid intimidating participants by the domain itself. Third, its real-world origin increased the external validity of our study. Finally, it has been used successfully in other studies (e.g., [Beckwith et al. 2007]), which not only provided evidence as to suitability for spreadsheet users in a lab setting, but also allowed us to harvest the bugs made by previous participants for use in the current study. We seeded the spreadsheet with a total of ten real bugs harvested from the spreadsheets of Beckwith et al.'s participants, who were Seattle-area adult spreadsheet users from a variety of occupations and age groups. None had computer science training or occupations, but all were experienced Excel users. Hence, the bugs we harvested from their work were realistic in terms of the kinds of bugs real end users generate.

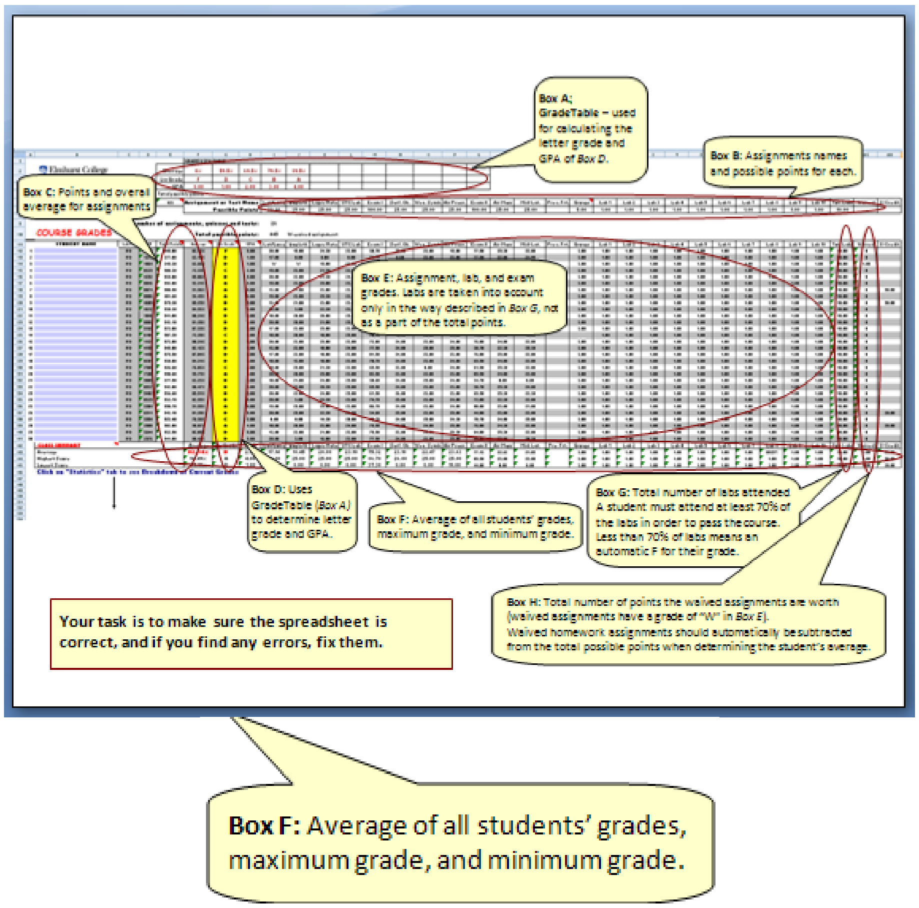

Box A; GradeTable – used for calculating the letter grade and GPA of *Box D*.
Box B: Assignments names and possible points for each.
Box C: Points and overall average for assignments
COURSE GRADES
Box E: Assignment, lab, and exam grades. Labs are taken into account only in the way described in *Box G*, not as a part of the total points.
Box D: Uses GradeTable (*Box A*) to determine letter grade and GPA.
Box F: Average of all students' grades, maximum grade, and minimum grade.
Box G: Total number of labs attended A student must attend at least 70% of the labs in order to pass the course. Less than 70% of labs means an automatic F for their grade.
Your task is to make sure the spreadsheet is correct, and if you find any errors, fix them.
Box H: Total number of points the waived assignments are worth (waived assignments have a grade of "W" in *Box E*). Waived homework assignments should automatically be subtracted from the total possible points when determining the student's average.
Box F: Average of all students' grades, maximum grade, and minimum grade.

Figure 2. (Top) A thumbnail of the description handout of the grade-book spreadsheet. (Bottom) Blowup of description Box F.

We harvested a variety of bug types from these previous users. Six of the harvested bugs were formula inconsistencies with respect to other formulas in the same row or column. For example, some cells' formulas omitted some students' grades in calculating the class average. Three more bugs were logic errors that had been propagated by their original authors over the entire row or column (e.g., using the ">" operator instead of ">="). The last bug was not part of any group of similar formulas: it counted lab attendance as a part of the total points, but should not have done so. This collection of ten real end-user bugs provided variety in the types of bugs our participants would need to track down and fix.

3.4 Spreadsheet Environment and Tutorial

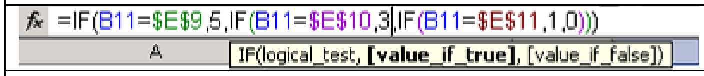
The environment for the study was Excel 2003. To make sure participants were familiar with some of Excel's debugging features, we pointed the "auditing" collection out to them in a pre-session tutorial and suggested that these features might help them make debugging progress. However, participants were free to use any Excel features they wanted.

The tutorial's wording was about features per se, and carefully avoided hints about how to use these features strategically. Table 2 summarizes the features we presented during the tutorial. The tutorial was hands-on: the participant used the features as the tutorial went along. In addition, since most participants had not used these features prior to this experiment, everyone had five minutes after the tutorial to try the features out on their own before moving on to the main task.

3.5 Analysis Methodology

We began by labeling the debugging state changes ("bug found", "bug fixed", and "reevaluating fix") in all eight of the participants' videos. Since these identifications did not require subjective judgments, only one researcher was

Table 2. The Excel 2003 debugging features, namely Excel auditing tools plus formula tool tips, covered in the "tour of features" tutorial.

Feature	Description
	The *Arrow Features* show the relationships between spreadsheet formulas. Left to right: Trace/Remove Precedent Arrows, Trace/Remove Dependent Arrows, and Remove All Arrows.
 =IF(B11=$E$9,5,IF(B11=$E$10,3,IF(B11=$E$11,1,0))) A IF(logical_test, **[value_if_true]**, [value_if_false])	*Tool tips* can be brought up by hovering over formulas to aid in their understanding.
	The *Evaluate Formula* tool allows users to see intermediate results, by observing how nested parts of a formula are calculated step by step.
	The *Error Checking* tool points suspicious formulas with a green triangle. These cells can either be stepped through in order or examined directly within the spreadsheet.
	The *Watch Window* tool allows users to watch one or more formulas and their results, which might be helpful when inspecting cells of interest which might have scrolled off the screen.

needed for this part. Sometimes participants believed they had fixed a bug when in fact they had not, so we also labeled these state changes as correct/incorrect. For example, editing a formula incorrectly was labeled "incorrect bug fix". Labeling these debugging state changes had two purposes. First, they pointed out milestones in the participants' success at the task as time progressed. Second, the count of participants' successful bug fixes was used to identify the corner cases for further analysis, namely the most successful and least successful female and male, a total of four participants.

For the four selected participants, two researchers then independently coded the videos according to the sensemaking codes to be described in Table 4 (which will be presented in Section 5), using the following procedure. First, each researcher independently coded 20 minutes of one of the four transcripts (about 10% of the total video data), using videos as well as written transcripts in order to have full access to the context in which actions were performed and words were spoken. The coders reached an 84% inter-rater reliability, calculated using the Jaccard index. Given this level of agreement, the two researchers then split up the remaining videos and coded them independently.

4. RESULTS: PARTICIPANTS' SUCCESS AT DEBUGGING

To provide context for the remainder of the results, we begin with the participants' success levels. Table 3 shows each participant's number of bug-find, bug-fix, and fix-reevaluation actions. We defined a bug-find action as

Table 3. For the bugs (10 in total) that needed fixing, these are the participants' successful and unsuccessful find and fix actions, as well as how many times they reevaluated the fixes they attempted. The top-scoring and bottom-scoring participants we selected for further detailed analysis are highlighted.

Participant	Bug Finds		Bug Fixes		Evaluations of Fixes
	Correct	Incorrect	Correct	Incorrect	
P05211130 (Female)	9	0	6	0	5
P05220830 (Male)	8	0	6	8	8
P05281130 (Female)	6	2	5	2	5
P05270830 (Female)	5	0	4	1	4
P05211600 (Male)	5	0	2	2	1
P05221230 (Male)	4	0	2	4	3
P05290830 (Female)	3	0	1	7	2
P05200830 (Male)	1	1	0	1	5

identifying a seeded incorrect formula as being faulty, a bug-fix action as changing a faulty formula, and a fix-evaluation action as checking a bug-fix action. We also used an “incorrect” modifier for the bug finds and bug fixes. Specifically, when the participant mistakenly identified a correct formula as being faulty, we labeled it as an incorrect bug-find, and when a participant edited a formula in a way that left the formula incorrect, we labeled it an incorrect bug-fix.

As Table 3 shows, participants’ sensemaking about where the bugs lurked was more successful than their sensemaking about how to fix those bugs. Specifically, in finding the bugs, six out of eight of the participants made no mistakes, and the remaining two made only one or two mistakes. When it came to actually fixing the bugs, only three of the participants made more correct fixes than incorrect ones, and one participant’s incorrect fix count was as high as eight.

The number of reevaluations averaged to less than one reevaluation per fix (51 fixes and 33 reevaluations). This is consistent with prior work that suggested reevaluation in debugging tends to be undersupported in spreadsheets, and users may believe that the immediate recalculation feature is sufficient for reevaluation purposes (a “one test proves correctness” view) [Wilcox et al. 1997].

To investigate end users’ sensemaking about spreadsheet correctness, we selected a subset of the participants to examine in detail. We chose the four most extreme participants two with the most bugs fixed (one female and one male), and the two with the fewest bugs fixed (one female and one male). These participants correspond to the shaded rows in Table 3, and the remainder of this paper focuses on them. From now on, we will refer to these four as SF (successful female, participant P05211130), SM (successful male, participant P05220830), UF (unsuccessful female, participant P05290830), and UM (unsuccessful male, participant P05200830).

5. RESULTS: THE SENSEMAKING MODEL FOR END-USER DEBUGGERS

5.1 Three Sensemaking Loops

Pirolli and Card characterized intelligence analysts’ sensemaking in terms of a major loop and its subloops, reflecting the iterative nature of sensemaking. Our data echoed this iterative character, but our participants’ subloops were organized under not one but *three* major loops that corresponded to different classes of challenges in our participants’ work. Our model is presented in Table 4.

We term the major loop, in which participants reasoned about the bugs and spreadsheet formulas/values, the Bug Fixing Sensemaking Loop; the loop in which they reasoned about the environment, the Environment Sensemaking Loop; and the loop in which they reasoned about common-sense topics and/or the domain, the Common Sense and Domain Sensemaking Loop. Considering these loops separately provided the conceptual benefit of focusing on the challenges introduced by a particular programming environment or domain separately from the challenges introduced by the difficulties of debugging that are independent of the environment or domain.

The Bug Fixing Sensemaking Loop was where our participants spent most of their time. We devote the next subsection to it.

The Environment Sensemaking Loop arose multiple times for all four participants. It was triggered when participants tried to make sense of Excel features or syntax. For example:

SF: “So, I’m clicking on the trace dependents. [Excel displays a small thumbnail of a table with an arrow pointing from it to the formula.] [Participant hovers over the little table and then tries clicking on it. Nothing happens.] And it goes to wherever… There’s a little box, but I don’t know what that means.”

These visits to the Environment Sensemaking Loop were sometimes disadvantageous, but other times led to leaps forward. We will point to examples of both in upcoming sections.

The third sensemaking loop was the Common Sense/Domain Sensemaking Loop. This loop involved reasoning about general knowledge items, such as trying to remember mathematical principles, or conventions used in the domain such as trying to recall how grades are usually computed. This loop was less common with our participants, perhaps because, as college students, they were very familiar with grade computations. Here is an example of accessing this loop:

SM: “In the grading, it says students must attend 70% of the labs in order to pass the course. Are, uh, is it possible to be waived from the labs?”

Table 4. A side-by-side view of how Pirolli and Card's Sensemaking model and definitions compare with our node-oriented sensemaking model of end-user debugging. Node numbers refer to those in Figure 1. Our model elements (the right hand column) were also our sensemaking code set; see text for details.

The Sensemaking Model for Analysts [Pirolli and Card 2005]	The Sensemaking Model for End-User Debuggers
External Data Sources (node 1)*:* All of the available information.	*External Data Sources:* Same as Pirolli/Card.
Shoebox (node 4)*:* Much smaller set of data, relevant for processing.	*Shoebox:* Data that a participant deemed relevant enough to "touch" in the spreadsheet or study environment. Examples: Particular cells selected, spreadsheet description handouts read, menus of features perused, help documents accessed, etc.
Evidence File (node 7)*:* Even smaller set of data extracted from the shoebox items.	*Evidence File:* Extracted from the shoebox, data that attracted a participant's interest enough for follow-up. Example: Wanting to find out more information about a suspicious cell.
Schema (node 10)*:* A large structure or overview of how the different pieces of data from the evidence file fit together: a re-representation of the data.	*Schema:* A structure or pattern a participant noticed as to how cells or information related. Examples: Declaring that all cells in an area were behaving properly or that a cell(s) did not fit the pattern.
Hypotheses (node 13)*:* A tentative representation of the conclusions with supporting arguments.	*Hypothesis:* A tentative idea about how to fix a particular bug based on the participant's schema. Example: "So it's saying that the group average is higher than it really was. I would say that is a mistake, since the formulas below it include all of them, this formula should include all" (Participant SF).
Presentation (node 16)*:* The work product.	*Presentation:* The work product. Example: An edit to fix a formula (the edit could be right or wrong).
Reevaluate (edge 15, from node 16 to 13)*:* After the presentation has been created, checking to make sure that the Presentation is indeed accurate.	*Reevaluate:* After changing a formula, making sure that the change was in fact correct. Examples: Trying to input different value to see the result of the newly edited formula, reviewing the formula to evaluate its correctness.
	Environment Sensemaking: A sensemaking loop for figuring out the environment. Examples: Trying to understand Excel's error triangle feature or formula syntax.
	Common Sense and Domain Sensemaking: A sensemaking loop for answering questions about common-sense and domain questions. Example: Trying to figure out how weighted averages are normally computed.

Because they are peripheral to our main research questions, we did not perform detailed analyses of the Environment and Common Sense/Domain Sensemaking Loops. However, we did code the instances of these loops' presence so that we could see the interactions between these loops and the Bug Fixing Sensemaking Loop.

5.2 The Bug Fixing Sensemaking Loop

In the Bug Fixing Loop, participants gathered information about the spreadsheet logic and made sense of it in order to create the final product, namely a bug fix. We derived the elements of our Bug Fixing Loop directly from the Pirolli/Card model for intelligence analysts [Pirolli and Card 2005]. We chose the Pirolli/Card model over the other sensemaking models presented in Section 2.2 because of Pirolli/Card's low-level focus on how data are used to bridge a gap. This low-level focus on Dervin's "bridge" aspect, combined with the high-level overview of the entire problem-solving process, mapped well to our investigation of end-user programmers' debugging processes.

Pirolli and Card characterized their model as consisting of four high-level sensemaking steps: information gathering, schematic representation of the information, development of an insight through manipulation, and the creation of a knowledge product. These steps clearly apply to the end-user debugging task. *Information gathering* involves finding data relevant to the task at hand by, for example, identifying relevant information on the handout or locating formulas and values relevant to a bug. An example of *schematic representation* is building a comprehensive picture of how multiple parts of the spreadsheet work together. An example of *development of an insight* is realizing the significance of a particular unexpected output value. Finally, the primary *knowledge product* is a formula modification intended to fix the bug.

Given this correspondence, the elements of the Bug Fixing Loop in our model mapped directly from the nodes from the Pirolli/Card model. All the data representation steps of the Pirolli/Card model are nodes; these are steps 1, 4, 7, 10, 13, and 16 in Figure 1. Given the complete set of nodes, the Pirolli/Card edges connecting neighboring nodes (representing mental activities that connect these nodes) are implicit. Therefore, the only edge we included explicitly was step 15 (Reevaluate), since reevaluation of changes has long been reported to be fundamental to debugging (e.g., [Nanja and Cook 1987]).

Table 4 shows our model's correspondence with Pirolli/Card's model. Note that the exclusion of the edges (except for step 15) simplifies our model. Excluding the edges had no real disadvantage because edges are implicit in node changes—to get from one node to another, one must traverse the edge connecting them. The nodes represent *data* with which a user works (such as the "shoebox"), not the *process* by which the user works with that data (such as "skimming"). The advantage of this data-oriented model was that the resulting code set greatly facilitated analysis: it was much easier for researchers to reliably (i.e., with high agreement) identify the data representation with which a participant was working than to reliably identify the process a participant was using. The right column of Table 4 thus served as our code set (except the top row which was participant-independent and therefore not of interest to our research questions). We used this code set according to the methodology previously described in Section 3.5.

Figure 3 shows thumbnails of the participants' progressions through the sensemaking steps up and down the Bug Fixing Sensemaking Loop. (Full-sized versions of these graphs will be shown later in Figure 5.) The graphs show participants "climbing" the Bug Fixing Sensemaking Loop, and then dropping down to earlier steps of the model. Note the prevalence of traversing adjacent nodes in the Bug Fixing Sensemaking Loop upward in direct succession. For example, participants often advanced from adding to the evidence file (yellow) to structuring that information into a schema (orange).

Exceptions to the forward progressions through consecutive steps of the Bug Fixing Loop were sometimes due to switches to the other two loops (Environmental or Common Sense/Domain). In the thumbnails, these loop switches are simply shown as gaps (white space in Figure 3), such as in Participant SF's second half. Another exception was steps backward through the sensemaking model, sometimes returning to a much earlier step in the process, as we shall see in more detail shortly.

6. RESULTS: SENSEMAKING MODEL TRAVERSAL STYLES AND STRATEGIES

6.1 Dominance of Foraging during Sensemaking

Figure 4 shows the sensemaking traversal frequencies for each sensemaking node in the Bug Fixing Loop, with separators marking the major subloops of the model. Left of the separators are the nodes Pirolli and Card grouped into the "foraging subloop," in which people search for information and classify information related to their task at

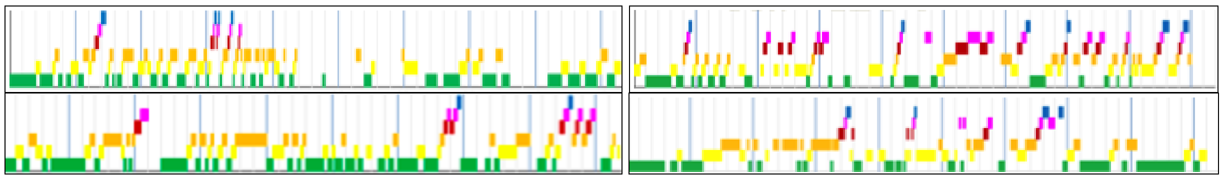


Figure 3. These thumbnails show the participants' upward climbs and downward drops in Bug Fixing sensemaking steps. X-axis: time. Y-axis: the step in the Bug Fixing sensemaking loop, from Shoebox to Reevaluate. (Time spent in the Environment and Common Sense/Domain Loops appear as horizontal gaps.) Top left: Participant SF. Top right: Participant SM. Bottom left: Participant UF. Bottom right: Participant UM.

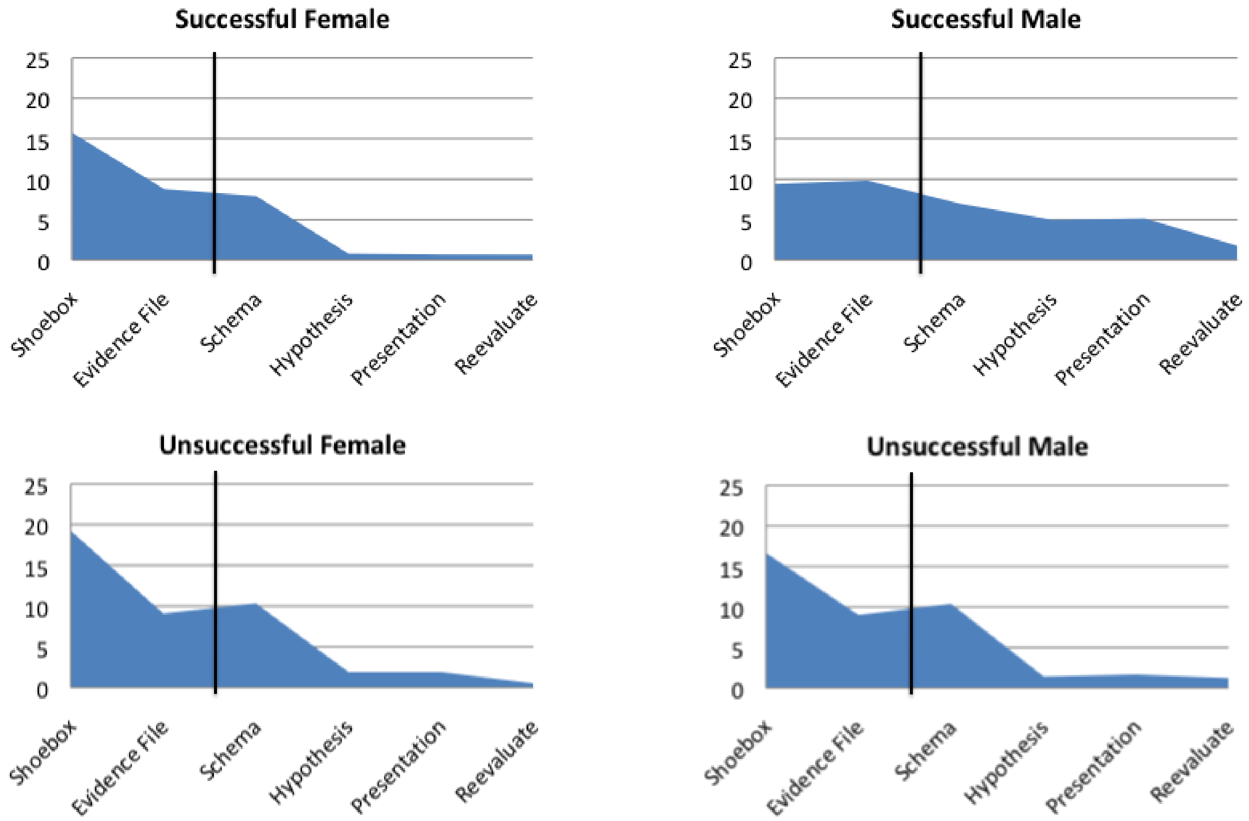


Figure 4. The amount of time in minutes (y-axis) spent at each Sensemaking model step (x-axis) by the participants. The vertical bars separate the information foraging subloop (left of the bars) from the sensemaking subloop (right of the bars).

hand. Right of the separators are the nodes of the Pirolli/Card "sensemaking subloop," in which people organize and make inferences from the information they have collected [Pirolli and Card 2005].

Information foraging has an associated theory of its own, termed information foraging theory [Pirolli and Card 1999]. The theory is based on optimal foraging theory, which describes how predators (animals in the wild) follow scent to a patch where the prey (food) is likely to be. Applying these notions to the domain of information, information foraging theory predicts that predators (people in need of information) will follow scent through cues in the environment to the information patch where the prey (the information itself) seems likely to be. Information foraging theory has primarily been used to understand web browsing, but also recently has been applied to understanding and predicting professional developers' code navigation during debugging [Lawrance et al. 2008].

Figure 4 reveals two interesting points about information foraging. First, although the information foraging part of sensemaking consists of only two steps, those two steps alone accounted for half to two-thirds of all four participants' time!

Their use of foraging was to gather information about how spreadsheet cells and formulas were working and how they interrelated. Everyone began this way. For example, Participant UF used the "Evaluate Formula" tool (Table 2) early in the session to look through numerous formulas and figure out how they worked.

An example illustrating promotion from Shoebox to Evidence in the foraging subloop was Participant SM's identification of cell G12's "strange"ness. A second example was Participant UF's identification of F12 and cells like it as being of interest and decided to gather new information to pursue them.

SM: [pauses] "Interesting. I think that G12 is a strange one. None of these students had special considerations or anything, right?"

UF: "If true, that tells the percentage. And if F12 is... Oh I bet those mean what the actual percentage is... [referring to symbols she was having difficulties figuring out] I'm going to look at the trace buttons to figure out where everything is coming from."

The second point that can be seen in Figure 4 is the remarkable similarity among three of the participants' (SF, UF, and UM) allocation of time. Also note how much their sensemaking style was dominated by the foraging subloop. In contrast, Participant SM's style was somewhat more evenly distributed across sensemaking steps. Note that both styles were associated with successful debugging in that both were styles of participants who were quite successful.

6.2 Sensemaking and Systematic Information Processing

The Selectivity Hypothesis [Meyers-Levy 1989] offers an explanation for the differences between the two types of sensemaking loop traversals used successfully. Recall that the Selectivity Hypothesis predicts that females will gather information comprehensively: getting a comprehensive understanding before proceeding with detailed follow-up. The Selectivity Hypothesis also predicts that the males will be more selective in the information they gather, such as tending to follow up on a salient cue right away. Meyers-Levy terms this style "heuristic processing," but because that style is characterized as being selective, we will refer to it as "selective processing." Note that neither style is implied to be better than the other; rather, the distinction is that the former is less selective than the latter as to which information to process.

Meyers-Levy proposed the Selectivity Hypothesis to describe people when working systematically. The Merriam-Webster Dictionary defines a *systematic* approach as "a methodical procedure or plan marked by thoroughness and regularity." The concept of a plan is closely related to the concept of *strategy*, defined by Merriam-Webster Dictionary as a "plan devised or employed toward a goal." Both successful participants indeed demonstrated strategic thoroughness and regularity, but the unsuccessful participants did not, and therefore fall mostly outside the scope of the Selectivity Hypothesis, as the next few paragraphs explain.

As Figure 5 helps to show, Participant SF spent most of her time in the foraging subloop viewing all cells in context, gathering Shoebox data (green, lowest row) and organizing it into Evidence (yellow, next row up). She appeared to place newly collected information into the context of the overall spreadsheet and of her other data gathered, as evidenced both by the regularity of occurrence and the length of time she spent in the Schema step (orange, third row from the bottom). Also the content of her utterances during these moments expressed her views of the role of each part of the evidence:

> *SF: "And, what else do we have? <looks at the description handout> So we checked all the bottom rows. And... <looks at screen> We checked to make sure [the area of grades] was hardcoded up at the top, to remain consistent, not going off formulas. So, and... Let's look in the section for class averages. Class summary."*

She seemed to have a threshold of "enough" information before moving beyond the Schema step to act upon it, as evidenced by the fact that she fixed bugs mostly in a batch, after having collected and processed much of the available information first. The only bug she fixed immediately upon finding it was an obvious fix, and did not interrupt her information gathering for long. Her approach worked well for her: recall from Table 3 that she correctly found nine bugs (more than any other participant), fixed six of them correctly, and had no incorrect fixes.

However, the comprehensive process also held disadvantages for Participant SF. Her method in the case of uncertainty was to gather more information. For example, when she thought one of the formulas looked odd (the second correct find, marked with a cross, in Figure 5's SF graph, at minute 12) but all of the cells within that region seemed equally incorrect, she did not pursue the fix right away, but rather continued with comprehensive information gathering. A disadvantage manifested itself when she did not mark the formula in any way for follow-up, and ultimately neglected to return to it. In addition to forgetting about that bug she found but never fixed, another disadvantage of comprehensive processing for Participant SF was that she did not abandon her comprehensive approach when it ceased to help her make progress. Instead, during the second half of the task, she spent most of her time following Excel's Error Checking feedback about where bugs might lie. She stayed with her comprehensive traversal through all 202 of Excel's "green triangle" warnings, even after spending over 10 minutes in this loop with only one bug find resulting. She did not attempt to fix this bug either, appearing to again rely on her memory of where the bug was, and ultimately did not follow up on a fix for it either. Instead, she opted to keep going comprehensively for the remaining 12 minutes of the task, during which time she found one more bug just before the time limit was reached.

In contrast to Participant SF, Participant SM was selective as to which information he gathered. He foraged only until he found a new bug to work on, at which point he narrowed his interest to trying to fix that bug, by moving up from the foraging subloop to the sensemaking subloop to Presentation and Reevaluation. For example, Participant SM found the same bug Participant SF found at minute 12; it was the second bug both of them found. But unlike Participant SF, he followed up on this bug right away. This happened to be the most difficult of the ten bugs to fix,

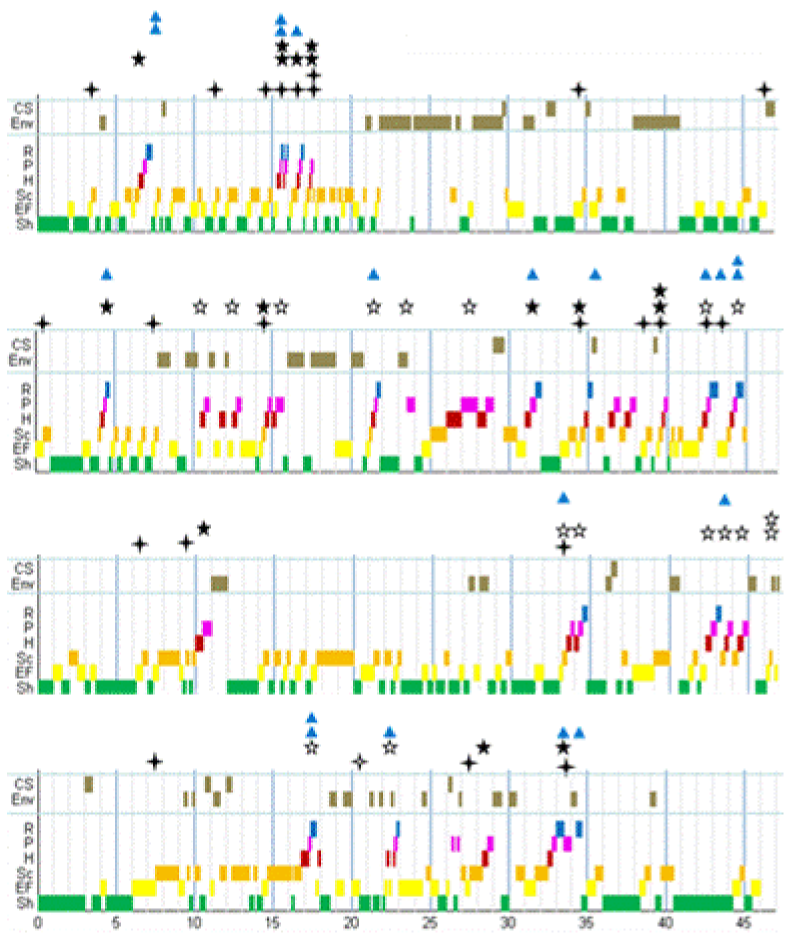


Figure 5. Full-sized view of the Sensemaking steps (y-axis) performed by four participants (from top to bottom: SF, SM, UF, UM) over time (x-axis). The top of each graph also shows the number of correct and incorrect finds and fixes, as well as the reevaluated fixes. ✦=correct (filled) or incorrect (hollow) finds; ★=correct (filled) or incorrect (hollow) fixes; ▲=reevaluates. UM has more correct finds and fixes in this figure than in Table 3, because UM introduced two bugs along the way that he later found and fixed. (The table refers only to the 10 seeded bugs.) The other three participants did not introduce bugs.

but he continued to pursue it, spending a lot of time iterating on the Schema, Hypothesis, Presentation, and Reevaluation steps. He found this bug in minute 8 and fixed it in minute 32; during this time, as Figure 5 shows, Participant SM iterated through the sensemaking loop to reach the Presentation step nine times! In contrast, Participant SF, who gathered much more information up front, never iterated to the top of the sensemaking loop more than once for any bug fix. Participant SM's process worked well for him: he found eight bugs successfully and fixed six of them (including the bug with which participants had the most difficulty).

However, the selective processing style also held disadvantages for Participant SM. He missed much of the information that had enabled Participant SF to spot and fix several bugs early. Participant SF fixed six bugs during the first half of the task, compared to only two by Participant SM. Furthermore, comprehensive processing might have provided useful information in solving the difficult bug upon which he spent so much time, in addition to potentially helping to find and fix some of the other bugs more quickly.

We have pointed out how consistent the above details for the successful participants were with the Selectivity Hypothesis. This consistency was triangulated against other aspects of the data in multiple ways. First, the amount of time spent in each subloop (Figure 4, previous section) helps to confirm Participant SF's comprehensive style and Participant SM's selective style. Second, Participant SF's batch of several Presentation instances (bug fix attempts) together versus Participant SM's incremental timing of each Presentation instance (Figure 5) also helps to confirm Participant SF's comprehensive style and Participant SM's selective style. Third, consider the number of transitions between steps. Figure 6 traces participants' sensemaking paths through the sensemaking model. Notice Participant

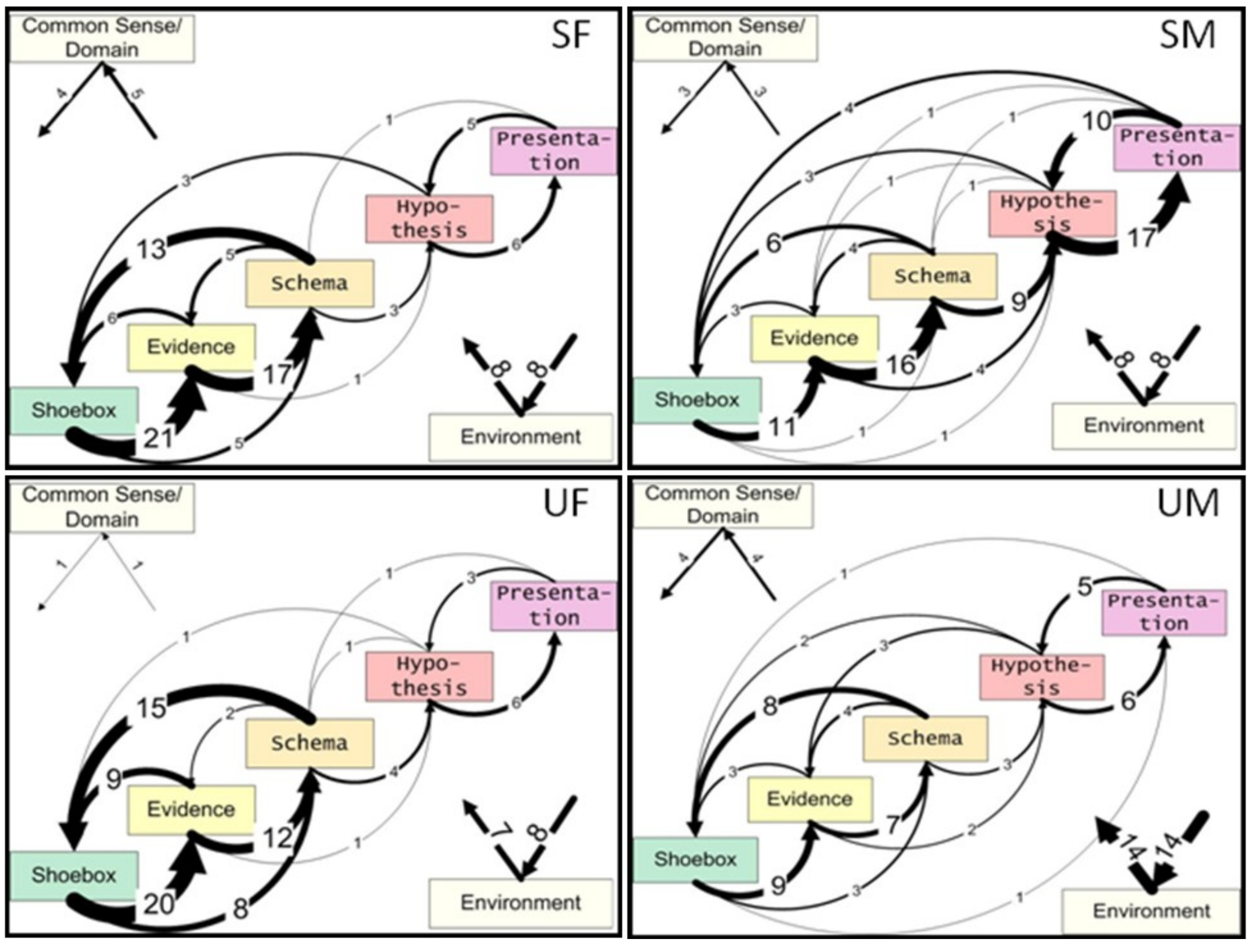


Figure 6. Frequency of transitions between sensemaking model steps by participant. Notice participant SF (top left) and participant UF (bottom left) transition mostly between different steps of the information foraging loop. Participant SM (top right) climbs up the sensemaking ladder in a mostly ordered manner, while participant UM takes a mixed approach and transitions into the Environment Loop more than any other participant.

SF's heavy emphasis on traversals between Shoebox to Schema, forming a cleanly separated "module" with only one transition between the "middle" of that subloop and bug follow-ups in the upper subloop. In contrast, Participant SM's most common transition was from Hypothesis to Presentation: his style showed a fairly uniform amount of activity on each upward transition progression in his pursuit of each bug, from Shoebox to Presentation and Reevaluation.

On the other hand, Participants UF and UM were mostly not systematic. One of these participants (UF) expressed plans but did not follow them; the other (UM) did not express plans at all. Further, neither showed signs of regularity or thoroughness. Instead, their approach seems better described as a sequence of counterproductive self-interruptions [Jin and Dabbish 2009].

We illustrate this first with Participant UF. Like Participant SF, Participant UF at first followed the layout of the specifications comprehensively (exhibiting regularity), but unlike Participant SF, she abandoned the comprehensive approach at her first bug find (minute 7). She focused on this bug (selective processing) for only three minutes, then found a second bug and chose to switch to that one instead (which she fixed immediately). This switch was productive in an immediate sense, but cost a loss of context regarding the first bug [Jin and Dabbish 2009]. Jin and Dabbish point out that triggered self-interruptions' disadvantages include difficulty refocusing on the first task's context, and likelihood of causing later self-interruptions. Indeed, at this point, about 10 minutes in, Participant UF's systematicness ended. For the rest of her session, her behavior was neither regular nor thorough: there ceased to be evidence of any "big picture" awareness, the focus of her verbalizations and formula reading shifted dramatically without closure on any one section or bug before moving on to the next, and her actions (cells selected for reading or editing) tended to be unrelated to the plans she did verbalize. Her behavior lacked the coverage to be considered comprehensive. Nor was it a systematic selective approach; for example, it was very different from Participant SM's selective but thorough focus on one bug, in which he always persevered with his most recent bug find until he had fixed it.

For example, although Participant UF occasionally expressed intent to follow up on one bug, she often immediately followed such verbalizations with actions *un*related to her expressed intent. For example, she expressed a plan at minute 18: "Okay, I'm gonna focus on the 'letter grades' <the first bug she found> because it seems like there is some inconsistency in how they are being calculated. Uhh, I'm not going to worry about the 'lab grades' because they all completed all of the labs. I'm going to ignore the 'total points' because it seems like those are all correct." This verbalization was, however, not followed by pursuing 'letter grades'; instead, she spent six minutes on 'total points' (which she had said she planned to ignore), and then seven minutes on 'GPA'. Following this, she briefly once again returned to the 'letter grade' formula, but for less than a minute, after which she moved to a new bug she then noticed in the 'waived' formula, never returning again to 'letter grade.' None of her activities after minute 12 led to any successful bug fixes.

Participant UM's approach was similar to Participant UF's but was even more ad hoc. Unlike Participant UF, who verbalized plans that she did not follow up on, Participant UM did not verbalize any plans at all. Many of his focus switches from one cell to the next were less than one second apart, far too little time to actually read a formula or warning message associated with that cell. "There was a, like, little green arrow thing next to D22. As I was looking down the list, and I just clicked on it. And then I just clicked on the error checking and..." This is in sharp contrast to the way Participant SF used the same Error Checking (green triangles) tool. When she used this tool, her verbalizations described use of the tool in the context of the whole spreadsheet, stating that she wanted to look at all of the inconsistent formula warnings in the spreadsheet (systematic comprehensive processing). When Participant UM used the same tool, his ad hoc behavior and quick attention switches suggest that the tool was instead primarily a self-interruption trigger.

Considering sensemaking from the standpoint of systematicness thus yields three classes of insights. First, for the two participants whose behavior was systematic, our data supports the Selectivity Hypothesis, with the female choosing a comprehensive information processing style and the male following a selective information processing style, just as the Selectivity Hypothesis predicts. Second, we observed several advantages and disadvantages with each systematic style. Third, the lack of systematicness of the other two participants helps us to understand why they ran into trouble and where. A "why" insight comes from the details revealing their numerous self-interruptions with attendant loss of context, and "where" insights are revealed by the graphs in Figures 4-6, which make clear that both unsuccessful participants spent a lot of time trying to build a Schema and also switched quickly in and out of the Environment Loop. These are two of the trouble spots we describe in more detail in the next section.

7. RESULTS: SENSEMAKING TROUBLE SPOTS

7.1 The Environment and Common Sense / Domain Loops

Recall that our sensemaking model has three loops: the main Bug Fixing Loop, the Environment Loop, and the Common Sense / Domain Loop. When participants exited the main Bug Fixing Loop to go into one of the other two loops, it was usually to go to the Environment Loop. Departures to the Common Sense / Domain Loop were few in number, and tended to be short in duration, but it is not surprising that our participants did not spend much time trying to make sense of the domain, since grade calculation is familiar to students.

Self-interruptions to switch to the Environment Loop arose in two situations. The first was when participants were having difficulties with some construct in the software (formula syntax, features, etc.). Participant UM had many instances of this situation, transitioning in and out of the Environment Loop almost twice as often as the other three participants (Figure 6). For example, while using the Evaluate Formula feature to understand a lengthy formula, he said:

> *UM: "And then, when I click this 'Evaluate Formula' button, it says if Z12 is greater… I forget what the name of the symbol… The greater than or equal to symbol. [Clicks Evaluate a couple of times] False. [Clicks Evaluate some more.] He gets an F. [Shakes head.] That doesn't make any sense."*

The other three participants also spent time trying to understand features' meanings and operators or functions they could use. For example:

UF: "I'm just trying to figure out again how to do an 'AND' statement. [Tries it and gets an error message.] Yeah, I figured that would happen."

SM: "Um, how would I assign a number to W? [Pauses.] Let me do a look-up formula. [Searches Help for VLOOKUP.]"

The second situation in which participants switched to the Environment Loop arose when they wanted the environment's suggestions on what to do next. An example of this was Participant SF's reliance on Excel's green triangles to lead her through suspicious cells in the hopes of finding more bugs. She spent about twice as many minutes in the Environment Loop as any of the other participants (SF: 10.8 min, SM: 5.8 min, UF: 3.7 min, UM: 4.2 min).

SF: "So the rest of them are correct. [Double checking that they're correct.] And, what else do we have? [Decides to follow up on Excel's green Error Checking triangles] <details omitted> Trying to figure out why… Hm… <details omitted> Why is that one [formula] inconsistent from the one next to it?"

Participant UM also tried to follow Excel's Error Checking feedback about what to do next, although with less success.

The graphs of the successful versus unsuccessful participants show a marked difference in their excursions into the Environment Loop (Figure 5). The two successful participants both tended to remain in the Environment Loop for longer periods at a time than the other two participants, cycling through it until they reached the goals that had sent them into the Environment Loop. The unsuccessful participants, on the other hand, tended to spend only short times in the Environment Loop, usually returning to the Bug Fixing Loop without a satisfactory answer. In these outcomes, participants' short times in the Environment Loop were simply interruptions, and did not deliver benefits to their debugging efforts.

7.2 Sticking Points Moving Up the Sensemaking Steps

As Figure 5 shows, instances in which participants made progress—with new bugs found or fixes at least attempted—were almost all marked by (1) *rapid* transitions between (2) *adjacent* steps (3) *upward* in the sensemaking model. The rapidity of the transitions during successful periods is apparent in Figure 5: the periods culminating in bugs found or fixes attempted were characterized by numerous tiny chunks of time spent in one step before moving to another sensemaking step. A close look at the figure shows not only the rapidity of transitions, but also that the transitions during these periods of progress were almost entirely between adjacent sensemaking steps, and were almost entirely upward.

Deviations from this pattern were usually signs of trouble. A case in point was the unsuccessful participants' propensity to get "stuck" on the Schema step. We were alerted to this possibility by the odd looking bumps in their graphs at the Schema step in Figure 4. In fact, as Figure 7 shows, during the first half of the task, the two unsuccessful participants were stuck at the Schema step for minutes at a time. There were no bug finds during these long stretches and, upon exiting the Schema step, the participants almost always went all the way back to the Shoebox (which can be seen in Figure 5), rather than progressing upward to Hypothesis or down to the adjacent Evidence File to reconsider the usefulness of data previously identified as being pertinent. Thus, the Schema step, in which participants synthesized the information they had decided was relevant evidence, was clearly a trouble spot for the two unsuccessful participants.

Transitions between some sensemaking steps may be detectable by tools. For example, attempts to fix bugs are, by definition, edits to cells that have formulas in them. Similarly, periods characterized by displaying several

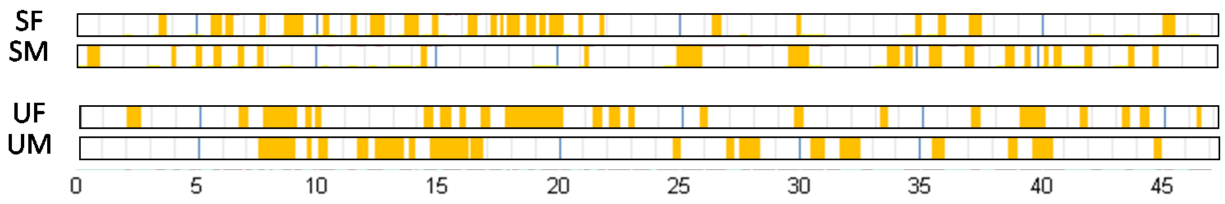


Figure 7. Excerpts from Figure 5 to bring out times spent in the Schema step. Note that the successful participants (top two) switched in and out of the Schema much more quickly than their unsuccessful counterparts (bottom two), who had a tendency to get stuck on that step.

different formulas may correspond to the Shoebox step, and periods characterized by reviewing formulas already viewed before may correspond to the Evidence step. If these kinds of detection can be automatically done, tools may be able to use this information to discern when a user is stuck at the Schema step and having trouble making debugging progress. This knowledge might then be used to focus on-line help or assistance tools that users access during these periods.

7.3 Moving Down the Sensemaking Steps

Although upward transitions tended to move incrementally, downward transitions were less predictable. In fact, participants had more than twice as many "step skips" in their downward transitions as they did in upward transitions. When moving in a downward direction, they most often fell all the way down to the Shoebox stage, as Figure 6 shows. A possible interpretation of these fallbacks to the beginning is that it may have seemed easier for participants to make progress based on newly collected data than to sort out which of the earlier steps led to a correct or incorrect conclusion.

Only one step was less subject to the "back to square one" phenomenon: the Presentation step. Recall that Reevaluate was a transition (edge) from the Presentation step down to the Hypothesis step, resulting in either the validation or rejection of the hypothesis. For all four participants, this step was the *only* step in which returning to the previous step dominated over going back to the beginning.

What happened from the Hypothesis step on, is still a mystery. The Sensemaking model might suggest that participants would then search for support for their hypothesis in the Schema step, perhaps judging which assumptions made at that step were correct and incorrect, determining whether to move up or down from there. However, the transition from Hypothesis back to the Schema was taken only *once* by Participant SM and Participant UF, and *never* by Participant SF and Participant UM. Thus, it appears that the participants neither incrementally changed nor revisited their Schema after the Hypothesis step.

8. DISCUSSION

The previous section's analysis makes clear the contrast in sensemaking traversal patterns between trouble spots versus instances of forward progress. This contrast suggests opportunities on how tools might detect the user's sensemaking step, which could lead to tools whose support is targeted at exactly that sensemaking stage.

For example, user accesses of help mechanisms were a sign of detours to the Environment Loop, and quick abandonment of a feature for which the user had just sought help would suggest that the detour was an unproductive one. This implies a need for the tool to explain the subject matter a different way if the user returns to the feature later. Other examples that could be detected by tools were long periods of formula inspections, which were usually in the Shoebox stage, and periods of follow-ups such as tracing formulas back through dataflow, which were often signs of the Evidence stage.

If tools like these were able to detect the user's current sensemaking step and compare it to the last few sensemaking steps, the tool might then be able to tailor its behavior depending on whether the user was progressing up the sensemaking loop versus systematically moving down versus falling down precipitously. For example, recall that often, the participants' downward transition patterns skipped many steps, thereby losing portions of sense already made, as with Participant SF who forgot about bugs she had already located and therefore never attempted to fix. Tools that could help users record and track evidence and hypotheses already gathered might be possible in a very low cost way, enabling users to systematically revisit otherwise forgotten or erroneously rejected assumptions. Just as a Sudoku player might recognize the usefulness of keeping track of penciled-in assumptions about which values are still viable for a square, and crossing them out one at a time, for end-user programmers the externalization of assumptions might help them notice important patterns and see interrelationships they may not have detected when keeping everything in the head. Two tool examples that allow tracking of one kind of assumptions in spreadsheet debugging are value range assertions for Forms/3 [Burnett et al. 2003] and Excel's data validation feature. Perhaps future, lighter weight tools are possible that allow tracking of other assumptions the user has made but might want to revisit.

One thing the model revealed was the dominance of information foraging. This aspect of sensemaking occupied half to two-thirds of participants' time; yet, foraging in end-user debugging has not yet been discussed in the literature on end-user programming practices. There is, however, recent research about professional programmers' debugging that has proposed tool possibilities based on information foraging theory: for example, constructs such as scent could be used to analyze the efficacy of environments, tools, and source code [Lawrance et al. 2008]. Our results also suggest the need for tools to explicitly support information foraging by end-user debuggers, in this case in spreadsheets, where theory constructs such as scent could be applied to spreadsheet formulas, layout, and structure.

Further, the model revealed that the two information processing styles proposed by others' research appeared to correspond to successful foraging, namely the comprehensive and selective styles. However, while both comprehensive and selective processing were successful styles, both also had disadvantages. For example, Participant SF lost track of bugs she had found while focusing on comprehensive processing, and Participant SM's bug finding and fixing seemed hampered by a lack of information. The lack of systematicness and its toll on the other two participants was far more obvious. Finally, although most of the participants attempted to traverse the spreadsheet systematically at least during some periods, they all missed some cells. These examples suggest that tools should facilitate systematic traversals of the spreadsheet, and further should do so in a way that is conducive to either the comprehensive or to useful selective styles of information processing, such as depth first.

As with all empirical studies, our study's threats to validity need to be taken into account in assessing our results. An internal validity threat in our study is that the specific spreadsheet or the specific bugs seeded could affect the participants' sensemaking process as they debugged. To reduce this threat, we harvested bugs that had been created as side effects of other work by experienced spreadsheet users working on the same real-world gradebook spreadsheet used in our study. A lack of understanding of Excel 2003 could also have influenced results, which we attempted to mitigate by requiring participants to be familiar with Excel 2003, (in fact, selecting the ten most experienced volunteers who responded to our recruitment notice), and giving a tutorial on certain features to ensure specific skill levels. Our study contains a construct validity threat because think-aloud data is known to be a highly imperfect representation of what humans actually think. We attempted to mitigate this threat by also collecting behavior data (actions), which were used for triangulating with spoken data. Regarding external validity, a think-aloud study is by definition an artificial situation that does not occur in the real world. Further, our participants may not be representative of the larger population of end-user spreadsheet users. In addition, lab experiments entail artificial constraints; in our case these included a tutorial before the task, a short time (45 minutes) to complete the debugging task, and presence of a computer-based video-recorder, any of which could have influenced participants' actions. These threats can be fully addressed only through future studies using different spreadsheets, bugs, and participants, and we hope future researchers will be interested in making use of our model for exactly this purpose.

9. CONCLUSIONS AND FUTURE WORK

This work represents the first application of sensemaking research to end-user debugging. To gain a sensemaking perspective on end users' debugging, we began by deriving a sensemaking model for end-user debugging. The model owes its roots to the Pirolli/Card sensemaking model for intelligence analysis, and from this start, we derived the new model for end-user debugging from our participants' data. The data revealed not just one major sensemaking loop, but three intertwined major loops, which we termed the Bug Fixing Loop, the Environment Loop, and the Common Sense / Domain Loop. We then used the model and its three major loops to shed light on end users' debugging approaches and problems that arose in the sensemaking central to debugging.

One contribution of this work has been (1) a new model of sensemaking by end-user debuggers, which consists of three connected sensemaking loops: one for reasoning about the "program" itself, one for reasoning about the programming environment, and one for reasoning about the domain. This model then enabled empirical contributions revealing (2) the dominance of information foraging in our end users' debugging efforts, (3) two successful strategies for systematic sensemaking and their consistency with gender difference literature in information processing styles, (4) a detailed account of transitions among sensemaking steps and among the three sensemaking loops in our model, (5) the sensemaking sequences that were tied to successful outcomes versus those that identified trouble, and (6) specific sensemaking trouble spots and consequent information losses. These findings suggest a number of opportunities for researchers and tool designers to consider how to support end users' sensemaking as they debug.

Regarding our own future work, we plan to do exactly that—experiment with tool ideas to better support end-user debuggers' sensemaking. We plan an intertwined set of further empirical work and tool experimentation. The purpose of these efforts will be to investigate how an end-user programming environment can better support end-user debugging by following opportunities and fulfilling needs the sensemaking model has helped reveal. The empirical component will not only inform our efforts, but also will help us ultimately to understand *whether* our new tools work and *how* tools might best support their sensemaking efforts of female and male end-user programmers. We believe that the fresh perspective the sensemaking model provides on the difficult task of debugging may hold the key to paving the way to a new generation of sensemaking-oriented debugging tools, and we hope other researchers will join us in these investigations.

ACKNOWLEDGEMENTS

We thank George Robertson for introducing us to sensemaking research, Peter Pirolli for his thoughts on our mapping of his sensemaking for intelligence analysts model to end-user debugging, and Joe Markgraf, Akshay Subramanian, and Rachel White for their help transcribing the participants' think-aloud videos. Laura Beckwith's background research into implications of the selectivity hypothesis on male and female problem solvers was also a key influence on this work. We especially thank the participants of our study. This work has been supported in part by the EUSES Consortium under NSF 0325273, by NSF 0917366, by AFOSR FA9550-09-1-0213, and by an IBM Faculty Award.

REFERENCES

ABRAHAM, R. AND ERWIG, M. 2007. A type system based on end-user vocabulary. In *Proceedings of the Symposium on Visual Languages and Human-Centric Computing*. IEEE, 215-222.

AYALEW Y. AND MITTERMEIR R. 2003. Spreadsheet debugging. In *Proceedings of the European Spreadsheet Risks Interest Group*.

BANDURA, A. 1986. *Social Foundations of Thought and Action*. Prentice Hall, NJ.

BATES, M. 1990. Where should the person stop and the information search interface start? *Information Processing and Management, 26*(5). 575–591.

BECKWITH, L. BURNETT, M., WIEDENBECK, S., COOK, C., SORTE, S., AND HASTINGS, M. 2005. Effectiveness of end-user debugging software features: Are there gender issues? In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*. ACM, 869-878.

BECKWITH, L. KISSINGER, C., BURNETT, M., WIEDENBECK, S., LAWRANCE, J., BLACKWELL, A., AND COOK, C. 2006. Tinkering and gender in end-user programmers' debugging, In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*. ACM, 231-240.

BECKWITH, L., INMAN, D., RECTOR, K., AND BURNETT, M. 2007. On to the real world: Gender and self-efficacy in Excel. In *Proceedings of the Symposium on Visual Languages and Human-Centric Computing*. IEEE, 119-126.

BOEHM, B. AND BASILI, V. 2001. Software defect reduction top 10 list. *Computer, 34(1)*, 135-137.

BURNETT, M., COOK, C., PENDSE, O., ROTHERMEL, G., SUMMET, J., AND WALLACE, C. 2003. End-user software engineering with assertions in the spreadsheet paradigm. In *Proceedings of the International Conference on Software Engineering*. IEEE, 93-103.

BURNETT M., COOK C., AND ROTHERMEL G. 2004. End-user software engineering. *Communications of the ACM 47(9)*, 53-58.

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

BUTLER, R. 2000. Is this spreadsheet a tax evader? How HM Customs and Excise test spreadsheet applications, In *Proceedings of the 33rd Annual Hawaii International Conference on System Sciences*, 6 pages.

BYRNES, J. P., MILLER, D. C., AND SCHAFER, W. D. 1999. Gender differences in risk taking: A meta-analysis. *Psychological Bulletin 125*. 367-383.

DERVIN, B. 1984. A theoretic perspective and research approach for generating research helpful to communication practice, *Public Relations Research and Education*, *1*(1). 30-45.

DERVIN, B., FOREMAN-WERNET, L., AND LAUNTERBACH, E. (Eds.). 2003. *Sense-making methodology reader: Selected writings of Brenda Dervin*. Cresskill, NJ: Hampton Press, Inc., 215-231.

EUSPRIG 2009. Spreadsheet Mistakes News Stories, European Spreadsheet Risks Interests Group site, http://www.eusprig.org/stories.htm, Accessed June 5, 2009.

FERN, X., KOMIREDDY, C., GRIGOREANU, V., AND BURNETT, M. 2009. Mining problem-solving strategies from HCI data. *ACM Transactions on Computer-Human Interaction* (to appear).

FINUCANE, M., SLOVIC, P., MERZ., C-K., FLYNN, J., AND SATTERFIELD, 2000. T. Gender, race and perceived risk: the white male effect. *Health, Risk and Society 2*(2), 159-172.

FISHER II, M. AND ROTHERMEL, G. 2005. The EUSES Spreadsheet Corpus: a shared resource for supporting experimentation with spreadsheet dependability mechanism. *1st Workshop on End-User Software Engineering*. 47-51.

FURNAS, G. AND RUSSELL, D. 2005. Making sense of sensemaking. *ACM CHI '05 Extended Abstracts on Human Factors in Computing Systems*. ACM, 2115-2116.

GALLAGHER A., DE LISI R., HOLST P., MCGILLICUDDY-DE LISI A., MORELY M., AND CAHALAN C. 2000. Gender differences in advanced mathematical problem solving, *J. Experimental Child Psychology 75*, 3, 165-190.

GRIGOREANU, V., BECKWITH, L., FERN, X., YANG, S., KOMIREDDY, C., NARAYANAN, V., COOK, C., AND BURNETT, M. 2006. Gender differences in end-user debugging, revisited: What the miners found. In *Proceedings of the Symposium on Visual Languages and Human-Centric Computing*. IEEE, 19-26.

GRIGOREANU, V., CAO, J., KULESZA, T., BOGART, C., RECTOR, K., BURNETT, M., AND WIEDENBECK, S. 2008. Can feature design reduce the gender gap in end-user software development environments? In *Proceedings of the Symposium on Visual Languages and Human-Centric Computing*. IEEE, 149-156.

GRIGOREANU, V., BRUNDAGE, J., BAHNA, E., BURNETT, M., ELRIF, P., AND SNOVER, J. 2009. Males' and Females' Script Debugging Strategies. *Second International Symposium on End-User Development*. Springer, 205-224.

HARTZEL, K. 2003. How self-efficacy and gender issues affect software adoption and use. *Communications of the ACM 46*, 9, 167-171.

IOANNIDOU, A., REPENNING, A., AND WEBB, D. 2008. Using scalable game design to promote 3D fluency: Assessing the AgentCubes incremental 3D end-user development framework. In *Proceedings of the Symposium on Visual Languages and Human-Centric Computing*. IEEE, 47-54.

JEFFRIES, R. A. 1982. Comparison of debugging behavior of novice and expert programmers. Paper presented in the AERA Annual Meeting. Pittsburgh, PA, Department of Psychology, Carnegie Mellon University.

JIN, J. AND DABBISH, L. 2009. Self-interruption on the computer: A typology of discretionary task interleaving. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*. ACM, 1799-1808.

KELLEHER, C. AND PAUSCH, R. 2005. Lowering the barriers to programming: A survey of programming environments and languages for novice programmers, *ACM Computing Surveys*.

KELLEHER, C., PAUSCH, R., AND KIESLER, S. 2007. Storytelling Alice motivates middle school girls to learn computer programming, In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*. ACM, 1455-1464.

KISSINGER, C., BURNETT, M., STUMPF, S., SUBRAHMANIYAN, N., BECKWITH, L., YANG, S. AND ROSSON, M. 2006. Supporting end-user debugging: What do users want to know? In *Proceedings of the Working Conference on Advanced Visual Interfaces*, ACM, 135-142.

KLEIN, G., MOON, B., AND HOFFMAN, R.R. 2006. Making sense of sensemaking 1: Alternative Perspectives. *IEEE Intelligent Systems 21*(4). IEEE, 70-73.

KO, A. AND MYERS, B. 2004. Designing the Whyline: A debugging interface for asking questions about program behavior, In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*. ACM, 151-158.

KO, A., MYERS, B., AND AUNG, HTET HTET 2004. Six learning barriers in end-user programming systems, In *Proceedings of the Symposium on Visual Languages and Human-Centric Computing*. IEEE, 199-206.

LAWRANCE, J., BELLAMY, R., BURNETT, M., AND RECTOR, K. 2008. Using information scent to model the dynamic foraging behavior of programmers in maintenance tasks. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*. ACM, 1323-1332.

LEEDOM, D. K. 2001. Final report: Sensemaking symposium. http://www.dodccrp.org/events/ 2001_sensemaking_symposium/docs/FinalReport/Sensemaking_Final_Report.htm. Command and Control Research Program, Office of the Assistant Secretary of Defense for Command, Control, Communications and Intelligence.

LITTMAN, D. C., PINTO, J., LETOVSKY, S., AND SOLOWAY, E. 1986. Mental models and software maintenance. In *Papers Presented At the First Workshop on Empirical Studies of Programmers on Empirical Studies of Programmers*. Ablex Publishing Corporation, NJ, 80-98.

MEYERS-LEVY, J. 1989. Gender differences in information processing: A selectivity interpretation. In P. Cafferata and A. Tybout, (Eds), *Cognitive and Affective Responses to Advertising.* Lexington Books, MA.

MYERS, B., KO, A., AND BURNETT, M. 2006. Invited research overview: End-user programming. In *Extended Abtracts of the SIGCHI Conference on Human Factors in Computing Systems*. ACM, 75-80.

NANJA, N. AND COOK, C. 1987. An analysis of the on-line debugging process. In *Empirical Studies of Programmers: Second Workshop*, Ablex Publishing Corporation, NJ, 172-184.

NARDI, B. A. 1993. *A small matter of programming: Perspectives on end-user computing*. MIT Press, MA.

O'DONNELL, E. AND JOHNSON, E. 2001. The effects of auditor gender and task complexity on information processing efficiency. *International Journal of Auditing 5,* 91-105.

PANE J. AND MYERS B. 1996, Usability issues in the design of novice programming systems, Carnegie Mellon University, School of Computer Science, *Technical Report CMU-CS-96-132*.

PANKO, R. 1998. What we know about spreadsheet errors. *Journal of End User Computing 10*(2). 15-21.

PANKO, R. AND ORDAY, N. 2005. Sarbanes-Oxley: What about all the spreadsheets? In *Proceedings of the European Spreadsheet Research Information Group*.

PHALGUNE, A., KISSINGER, C., BURNETT, M., COOK, C., BECKWITH, L. AND RUTHRUFF, J. 2005. Garbage in, garbage out? An empirical look at oracle mistakes by end-user programmers, In *Proceedings of the Symposium on Visual Languages and Human-Centric Computing.* IEEE, 45-52.

PIROLLI, P. AND CARD, S. 1999. Information foraging, *Psychology Review 106*(4). 643-675.

PIROLLI, P. AND CARD, S. 2005. The sensemaking process and leverage points for analyst technology as identified through cognitive task analysis. *Proceedings of the International Conference on Intelligence Analysis*. MITRE, VA.

POWELL M. AND ANSIC, D. 1997. Gender differences in risk behaviour in financial decision-making: An experimental analysis. *Journal of Economic Psychology* 18(6). 605-628.

PRABHAKARARAO, S., COOK, D., RUTHRUFF, J., CRESWICK, E., MAIN, M., DURHAM, M., AND BURNETT, M. 2003. Strategies and behaviors of end-user programmers with interactive fault localization. In *Proceedings of Symposium on Human-Centric Computing Languages and Environments*. IEEE, 15-22.

RODE J. AND ROSSON M. 2003, Programming at runtime: Requirements and paradigms for nonprogrammer web application development, In *Proceedings of Symposium on Human-Centric Computing Languages and Environments*. IEEE. 23-30.

ROMERO, P., DU BOULAY, B., COX, R., LUTZ, R., AND BRYANT, S.. Debugging strategies and tactics in a multi-representation software environment. *International Journal on Human- Computer Studies 61*. 992-1009.

ROSSON, M., SINHA, H., BHATTACHARYA, M., AND ZHAO, D. 2007. Design planning in end-user web development. In *Proceedings of the Symposium on Visual Languages and Human-Centric Computing*. IEEE, 189-196.

RUSSELL, D. M., STEFIK, M. J., PIROLLI, P., AND CARD, S. K. 1993. The cost structure of sensemaking. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*. ACM, 269-276.

SCAFFIDI, C., SHAW, M. AND MYERS, B. 2005, Estimating the numbers of end users and end user programmers, In *Proceedings of the Symposium on Visual Languages and Human-Centric Computing*. IEEE, 207-214.

SHRAGER, J. AND KLAHR, D. 1986. Instructionless learning about a complex device: the paradigm and observations. *International Journal of Man-Machine Studies* 25. 153-189.

STEFIK, M., BALDONADO, M., BOBROW, D., CARD, S., EVERETT, J. LAVENDEL, G., MARIMONT, D., NEWMAN, P., RUSSELL, D., AND SMOLIAR, S. 2002. The knowledge sharing challenge: The sensemaking white paper. http://www2.parc.com/istl/groups/hdi/papers/sensemaking-whitepaper.pdf.

SUBRAHMANIYAN, N., BECKWITH, L., GRIGOREANU, V., BURNETT, M., WIEDENBECK, S., NARAYANAN, V., BUCHT, K., DRUMMOND, R., AND FERN, X. 2008. Testing vs. code inspection vs. what else? Male and female end users' debugging strategies. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*. ACM, 617-626.

TORKZADEH, G. AND KOUFTEROS, X. 1994. Factorial validity of a computer self-efficacy scale and the impact of computer training. *Educational and Psychological Measurement 54*(3). 813-821.

WAGNER, E. AND LIEBERMAN H. 2004. Supporting user hypotheses in problem diagnosis on the web and elsewhere. In *Proceedings of the International Conference on Intelligent User Interfaces*, ACM, 30-37.

WILCOX, E., ATWOOD, J., BURNETT, M., CADIZ, J., AND COOK, C. 1997. Does continuous visual feedback aid debugging in direct-manipulation programming systems? In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*. ACM, 258-265.

AUTHOR STATEMENT

No part of this paper has previously been published or submitted anywhere: it is completely original. The related work references our earlier work on female and male end-user programmers' debugging. This is our first examination of end-user debugging's relationship to sensemaking models.