AN ABSTRACT OF THE DISSERTATION OF

James A. Hoag for the degree of Doctor of Philosophy in Mathematics Education presented on July 23, 2008

Title: College Student Novice Spreadsheet Reasoning and Errors.

Abstract approved: _____
Margaret L. Niess

The spreadsheet has become a common technology tool and is now a predominant form of end-user programming. Some of the same features that make spreadsheets excellent tools for ad-hoc development can introduce errors into the final product. Although a variety of research has been performed investigating methods to detect errors in spreadsheets, little has been done to investigate initial reasoning errors. The spreadsheet error taxonomy developed by Rajalingham, Chadwick, and Knight (2001) includes categories for reasoning errors that have not yet been investigated. Previous studies have categorized errors in existing spreadsheets, but have not analyzed the source of the error.

This study investigates the reasoning of college students while developing spreadsheets and examines the reasoning associated with errors generated in spreadsheet development. For this study, a phenomenological qualitative design incorporated a think-aloud protocol, interviews, and recordings of spreadsheet development of three purposefully-selected students. Data sources were analyzed to determine their reasoning, the types of errors produced based on the taxonomy, and associations between reasoning and errors.

The findings indicated that students used different types of reasoning in the mathematical phases of spreadsheet development than they do in the spreadsheet implementation phase. As novice spreadsheet developers, the students had significant difficulties translating problems into mathematical representations. Spreadsheet skills and concepts improved with practice through the course, but mathematical representations remained problematic. The students enjoyed using the spreadsheet as a tool for doing mathematical reasoning.

Several themes emerged as the study progressed: Reasoning differences during mathematical and spreadsheet phases of development; using icons for functions affected conceptualization of the functions; copy operations were perceived as "painting" rather than applying a formula to a series; and the effectiveness of the taxonomy for categorizing reasoning errors .

The results suggested modifications to student learning experiences leading to more accurate spreadsheet development: Integrate spreadsheets into mathematics courses; increase education in spreadsheet development; integrate formal design and testing components to the spreadsheet curriculum; include spreadsheet errors in the curriculum.

College Student Novice Spreadsheet Reasoning and Errors

by
James A. Hoag

A DISSERTATION

submitted to

Oregon State University

in partial fulfillment of
the requirements of the
degree of

Doctor of Philosophy

Presented July 23, 2008
Commencement June 2009

Doctor of Philosophy dissertation of James A. Hoag presented on July 23, 2008.

APPROVED:

Major Professor, representing Science and Mathematics Education

Chair of the Department of Science and Mathematics Education

Dean of the Graduate School

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

James A. Hoag, Author

ACKNOWLEDGEMENTS

I would like to express sincere appreciation to those who have made all this possible. I would like to thank my committee members, Dr. Larry Flick, Dr. Tim Budd, Dr. Alix Gitelman, Dr. Larry Enochs, and Dr. Emily van Zee. Also, thank you to Dr. Edith Gummer for the enthusiastic introduction into the Ph.D. program. I would like to express my gratitude to my advisor, Dr. Maggie Niess, for all the guidance, encouragement, and patience. Thanks to Dr. Ruey Shieh for sharing the journey and giving encouragement. I would like to thank my wife, Lee, for her continuous support and encouragement.

TABLE OF CONTENTS

TABLE OF CONTENTS (Continued)

TABLE OF CONTENTS (Continued)

LIST OF FIGURES

Figure Page

LIST OF FIGURES (Continued)

Figure Page

LIST OF TABLES

College Student Novice Spreadsheet Reasoning and Errors

Chapter I

Introduction

Current college students represent a group of individuals who have been exposed to computer technology and may have taken a course in technology literacy, including spreadsheets. There has been recent attention drawn to spreadsheet errors in industry, business, and research. Although many factors associated with errors in spreadsheets have been investigated, the reasoning involved has not been examined to a large degree. These college students will become spreadsheet end-users and developers and the learning experience will affect the quality of later spreadsheet applications they may develop. An investigation of student novice reasoning in spreadsheet development with a focus on reasoning that leads to errors may give insights into the causes of errors and methods to prevent them.

Since the advent of computers, enhanced technological capabilities have provided increased opportunities for solving problems. Computer software, such as the electronic spreadsheet, provides capabilities for solving application operations accurately and quickly. Using spreadsheets as cognitive tools, end-users have gained the capability to solve problems, develop models, and analyze data that previously were difficult or impossible. In the early days of computer solutions in mathematical problem solving, traditional programming languages were used. This process required specialized knowledge in program development. Today, electronic spreadsheets have replaced the need to rely on programming languages and have become the predominant method for

application programming where end-users have become spreadsheet programmers (Peyton Jones, Blackwell & Burnett, 2003; Panko, 1998). An end-user is someone who has a computational need and wants to make serious use of computers. End-user programming is the creation of an application that serves some function for the user (Nardi, 1993). A spreadsheet developer is a spreadsheet programmer. Note that an end-user may be someone who uses a spreadsheet application, but may also develop spreadsheets for their own use. As Grossman (2002) notes, these distinctions become difficult as someone who develops spreadsheets for other's use are end-user programmers but not end-users.

The extensive use of spreadsheets as critical decision-making tools has sparked interest in the reliability and correctness of these applications. Estimates indicate that as many as half of all spreadsheets that are created have errors, although this number may be as high as 90% (Panko, 2005). In those spreadsheets that contain errors, approximately 3% of the cells are in error, creating false results for the entire spreadsheet. Spreadsheets are rarely tested, and generally do not follow development guidelines (Panko, 2000). Spreadsheet developers often construct poorly designed and documented applications that do not lend themselves easily to modification (Isakowitz, Schocken, & Lucas, 1995). Yet, these spreadsheets are often used as if they are fully-documented applications. In reality, spreadsheet developers learn to create spreadsheets in ways that often lead to inaccurate, undependable, and inflexible applications.

People learn to develop spreadsheets in a variety of ways: formal courses, manuals, tutorials, peers. Baker, Powell, Lawson, & Foster-Johnson (2005) report that

only half of spreadsheet developers had formal training. In a formal course environment, learning may take place by emulating worked examples, engaging in a problem solving approach, and relying on instructor-led demonstrations, The goal of instruction is generally to develop spreadsheet skills and may include goals that students develop an understanding of the problem they are solving, the mathematical and spreadsheet concepts involved, and the solution process. However, in developing a spreadsheet, students may have mimicked an example to create a spreadsheet without understanding the underlying concepts. They may think they understand the problem and solution, but may misunderstand either or both. A common tendency is to believe the results are correct because the computer generated them (Panko, 2000). On the other hand, students may not take errors seriously, as there may be little impact in an academic or experimental setting. The motivation involved when developing spreadsheet applications may also be a factor in accuracy. Brown and Gould (1987) suspected experienced spreadsheet users in their study may not have been as concerned about errors as they would have been in a real-life situation

Most research of spreadsheet errors investigates testing and detection of errors. Little research exists on the correction or prevention of errors in spreadsheets (Panko, 2000; Burnett, Cook, Pendse, Rothermel, Summet, & Wallace, 2003). The literature on reasoning processes used during spreadsheet development is also scarce. Published efforts have focused on categorization of errors to identify areas for improvement and to understand the causes of errors. The identification of the causes for these errors is particularly important in determining possible methods of future correction or prevention.

Spreadsheet errors may result from not understanding the problems for which the applications are being developed, the capabilities of spreadsheets, or the mathematics underlying the solution. Spreadsheet end-users and developers also make accidental typing errors, mistakes in formulas, and errors in copying during the problem solving process; yet, without complete testing of the spreadsheet, the errors created by the spreadsheet developers often go undetected.

Often spreadsheet creators solve a problem based on a solution learned for a previous problem. In this case, they rely on analogical reasoning for the solution to a new problem. This analogical reasoning process may induce errors resulting from flaws in the reasoning process. For example, a student may have developed a solution for a particular problem. When presented with a similar problem, students may see the similarity and difference between the two problems and effectively develop an accurate spreadsheet representation of the new problem. Alternatively, they may simply try to mimic the spreadsheet approach used in a previous, possibly inaccurate problem solution. Or, an error might result by students not understanding how to apply the solution of the old problem to the new. On the other hand, they may not see the similarity between the two problems or not understand the relationships between them. In this case, they start a completely new process of development using a trial and error approach to solving the problem. Each of these situations has potential for introducing errors into the spreadsheet.

Statement of the problem

Research efforts to investigate spreadsheet errors have included classification of errors (Panko, 2005; Rajalingham, Chadwick, & Knight, 2001), testing spreadsheets (Panko, 2005; Rothermal, Cook, Burnett, Schonfeld, Green, &, Rothermal, 2000), adding design components to development (Janvrin & Morrison, 2000), and introducing elements of software engineering (Burnett et al., 2003; Rajalingham, Chadwick, Knight, & Edwards, 2000), and creating user-defined functions (Peyton Jones, Blackwell, & Burnett, 2003). Many of the proposed methods to reduce spreadsheet errors counteract the motivation for using the spreadsheet for solving particular problems. The advantages of the spreadsheet over traditional programming languages include flexibility, the ability to engage in ad-hoc development, and the ease of modification. Spreadsheet development is more closely related to the user's domain knowledge than programming languages due to apparently reduced abstractions that allow novices to quickly learn enough to develop an application to solve a specific problem. Errors are often introduced during a process that typically fails to include design or testing. Students tend to believe the spreadsheet results and have confidence in the solution without critically assessing the results (Panko, 2000). Panko feels this confidence is due to the phenomenon that since spreadsheet are easy to work with, developers feel they do well. The errors that are found informally only increase the confidence in the spreadsheet's accuracy. There may also be an effect due to overconfidence in computer-generated results. Also, the student developers are not as invested in the results as might be the case in an operational spreadsheet, one used in industry, business, or research on a routine basis.

Difficulties also arise during the modification of a spreadsheet, when a specific application is altered in order to solve other seemingly similar problems. In this case, the student may not have developed an accurate solution to the original problem or may have created a solution that is not amenable to changes. Many approaches to solutions impose restrictions on the flexibility and dependability of a broader application. Thus, as the spreadsheets are adapted, errors are generated, often unbeknownst to the student, the end-user. Furthermore, flaws in the solution may be propagated through the development process and maintenance of the solution. Determining more about how and why these errors are created in the spreadsheet development may help to identify instructional strategies for guiding students as spreadsheet end-users to design more accurate, flexible, and dependable spreadsheets.

Some efforts to understand spreadsheet errors have focused on classifying types of errors (Panko, 2000; Rajalingham, et al., 2001). The classification scheme generally used for spreadsheet errors has resolved existing errors into two main categories, quantitative and qualitative. Quantitative errors are those errors that produce an incorrect value. In the taxonomy suggested by Panko (2000), subcategories of quantitative errors are *logical*, *mechanical*, and *omission*. *Mechanical* errors are those errors due to mistyping. *Logic* errors result from incorrect formulas and *omission* errors are those resulting from omitting part of the problem detail in the solution.

On the other hand, qualitative errors are those errors that currently give correct values but perhaps result in incorrect values with future manipulations of the spreadsheet (Panko, 2000). Examples of these types of errors are the result of some of these actions:

- Hard-coding of values that are actually variables in the solution. These values are embedded in formulas and not visible as changes are made in the spreadsheet.
- Misrepresentation of the text labels or missing text labels that provide explanations for the users of the spreadsheets.
- Formatting problems that misrepresent the spreadsheet solution.

While this taxonomy may be satisfactory for classifying errors in existing spreadsheets, the taxonomy is not effective in describing the reasoning that leads to the errors resulting in undependable and inflexible spreadsheets.

Rajalingham, Chadwick, and Knight (2001) describe a more extensive taxonomy of errors that includes categories to classify reasons for the errors (see Figure 1.1). In this classification system, quantitative errors are those that lead to bottom-line numerical inaccuracy, as in the taxonomy used by Panko. However, the subcategories provide for finer classification. Quantitative errors are described as either *accidental* or *reasoning* errors. *Accidental* errors are mistakes and slips caused by negligence, such as typing errors. *Reasoning* errors are those that are created by entering incorrect formulas, either choosing the wrong algorithm or creating the wrong formula for implementation of the algorithm planned in the spreadsheet solution.

Subcategories of *reasoning* errors are *domain knowledge* and *implementation*. *Domain knowledge* errors are due to a lack of the understanding of the problem needed in order to accurately model the problem solution. Included in this classification are errors resulting from selection of an incorrect algorithm (*real-world knowledge*) and inaccurate

construction of a formula to implement a correctly chosen algorithm (*mathematical representation*). *Implementation* errors are due to lack of knowledge of the full use of the functions and capabilities of the spreadsheet package in use, with an understanding of the spreadsheet principles, concepts, constructs reserved words, and syntax. *Implementation* errors are further divided into *syntax* errors and *logic* errors. *Syntax* errors occur when the formula contains characters and symbols that are not recognized by the spreadsheet for performing the desired function. Most spreadsheets have the capabilities for signaling these errors. A *logic* error produces incorrect values resulting from an incorrect construction of a formula; this error may be due to a lack of understanding of the specific features and functions of the spreadsheet and thus produces an incorrect value.

The taxonomy originally developed by Rajalingham, Chadwick, and Knight (2001) original taxonomy has subsequently been revised (Figure 1.2) and suggestions for further revisions have been proposed. Rajalingham (2005) eliminated the *System* category as beyond the scope of research. He also revised the *Qualitative* subcategories. Purser and Chadwick (2006) suggest collapsing the *Accidental* categories since the user is often the developer. The *Reasoning* categories remain the same and, as yet, have not been investigated.

The process of developing a spreadsheet application from a written or verbal description of a situation or problem may include elements of problem solving. Research on problem solving indicates that several methods of reasoning strategies can be involved in the process, including means-end, trial-and-error, and analogical reasoning (Charney, Reder, & Kusbit 1990; Gick, 1986). Means-end analysis is a goals-based strategy where

the subject focuses on the goal and eliminates differences between the current state and the goal state. End-users might try different menu items or choose formulas that seem likely using a trial-and-error methodology. When using analogical reasoning, a user bases the solution on a new problem or a solution on an existing problem.

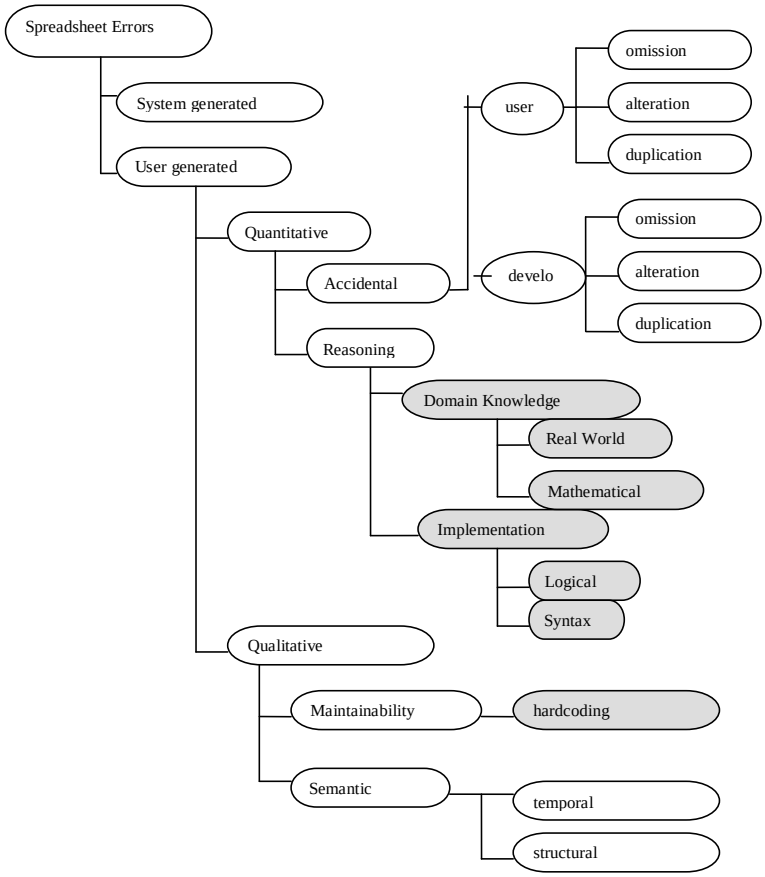


Figure 1.1 Taxonomy of spreadsheet errors (Rajalingham, Chadwick, & Knight, 2001). The gray shaded areas are those of interest in this study.

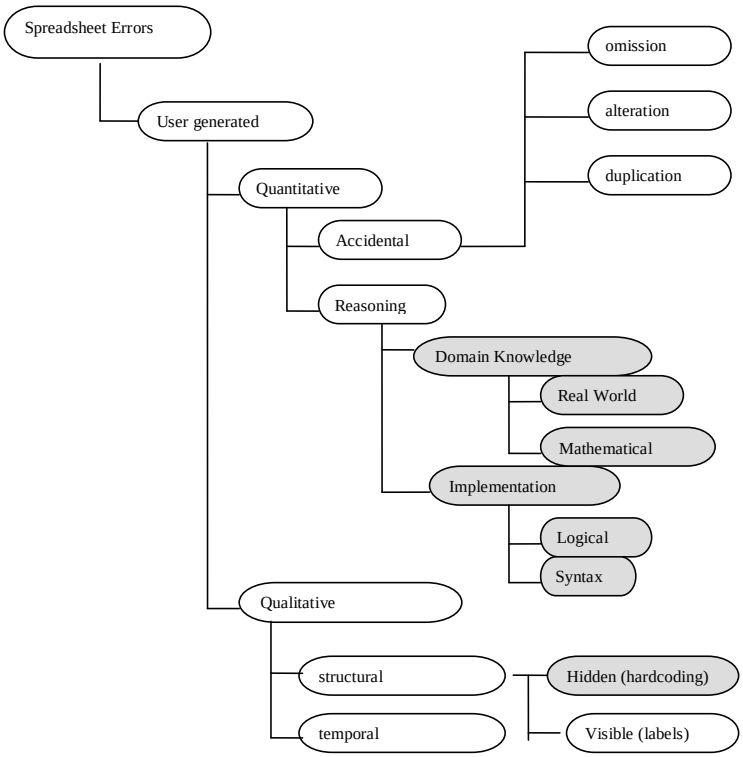


Figure 1.2 Revision of Rajalingham's (2005) revised taxonomy (Purser and Chadwick, 2006). The gray shaded areas are those of interest in this study.

If an end-user misunderstands the intermediate goals in means-end reasoning, errors may result. Using a trial-and-error approach is, of course, prone to errors. The processes identified as responsible for the reasoning errors (such as selection of an incorrect algorithm, inaccurate construction of a formula, semantic errors, and maintainability errors) are similar to errors that may result from analogical reasoning. Analogical reasoning is a common technique used in solving mathematical problems and

in creating computer programs. Mayer (2002) describes the analogical reasoning process (shown in Figure 1.3) as recognizing a base problem that is similar to the target, extracting the algorithm, and mapping the solution to the new problem.

The reasoning errors described by Rajalingham et al. (2001) may result during the analogical reasoning processes involved in spreadsheet development (Figure 1.4). *Domain knowledge* errors resulting from the lack of knowledge about the problem situation may occur during multiple stages in the problem solving process: recognizing the problem, abstracting the problem (selection of the wrong algorithm), and mapping the problem (inaccurate construction of a formula to implement a correctly chosen algorithm). *Implementation* errors due to a lack of knowledge on the use of functions and capabilities of the spreadsheet may occur during the process of mapping the solution of the old problem to the new problem.

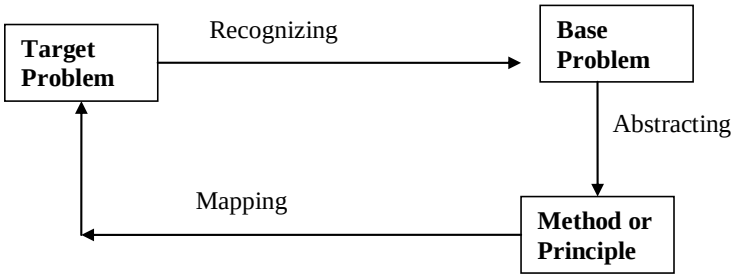


Figure 1.3 Analogical reasoning process for solving problems

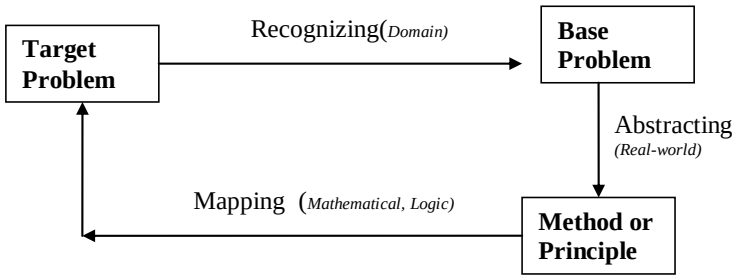


Figure 1.4 Spreadsheet reasoning and analogical reasoning

The errors described as reasoning errors by Rajalingham et al. (2001) are part of the quantitative error category, meaning the errors result in incorrect values. The taxonomy also includes categories for qualitative errors. The errors in the qualitative category are the same as those described by Panko (2000) earlier. These errors make understanding the spreadsheet difficult, increase the difficulty of maintenance, may result in errors after changes to inputs, or may become inaccurate if the spreadsheet does not allow for updates of variables. Data that have not been updated consistently are classified as *temporal* errors. *Structural* errors include flaws in the design or layout of the model, incorrect or ambiguous headings, and situations where assumptions are not reflected in the spreadsheet model. Visible errors of this type include headings and labels that are incorrect. *Hidden structural* errors are features of the spreadsheet that make it difficult to be updated or modified. A common example of a *structural hidden* error is hard coding of values, when a constant value is used in a formula rather than a cell reference. The result may be accurate for the value that was hard-coded, but does not provide an accurate model for other possible values. This error is a result of faulty reasoning or lack

or reasoning capability regarding spreadsheet functioning as the developer did not fully utilize the flexibility inherent in the spreadsheet paradigm. For the purposes of this study, maintenance errors are considered along with *implementation* errors, as these errors reflect incomplete or incorrect knowledge of the capabilities and constructs of spreadsheets.

People learn to develop spreadsheets in multiple ways: from co-workers, manuals, tutorials, trial-and-error, and formalized training. According to a survey conducted as part of the Spreadsheet Engineering Research Project (SERP) at the Tuck School of Business of Dartmouth College, only approximately one-half of spreadsheet developers have had any kind of formalized training (Baker, Powell, Lawson, & Foster-Johnson, 2005). Formal computer training is often heavily based on manual-type texts and tutorials, using worked examples and problem solving approaches (Charney, Reder, & Kusbit., 2001). Part of this training typically focuses on spreadsheet features and may or may not involve modeling of effective spreadsheet application development. Thus, the lack of a focus on solving problems using spreadsheets allows novices and often end-users to develop *ad hoc* models for specific single problems. Wong (2003) notes that when students learn traditional programming, they are taught to develop general solutions. Yet, with spreadsheet instruction, students are focused on finding solutions to specific problems, rather than more general solutions with the capability of solving multiple similar problems.

A review of several common spreadsheet texts reveals little attention to spreadsheet errors and debugging spreadsheets. The review indicates little or no content

on causes of spreadsheet errors, the impact of spreadsheet errors on problem solution, or methods for detecting errors in spreadsheets other than the available auditing utilities in a spreadsheet package. There is also little in the texts regarding testing spreadsheets for accuracy. The correct results are usually given in the textbook example, so a learner might develop a spreadsheet with the numeric values indicated, but the spreadsheet may not give accurate results for the range of possible inputs. Some texts do include an introduction to the auditing tools available in spreadsheets. These tools give graphical cues regarding inconsistencies in the spreadsheet. While some texts include design issues, these are largely focused on layout, rather than formulas.

The *reasoning* errors category and subcategories developed by Rajalingham et al. (2001) help explain the types of errors that may be generated during spreadsheet development. But little work has been done on how this inaccurate or accurate reasoning develops (Powell, Baker, & Lawson., 2008). An investigation of the reasoning used during spreadsheet development and in particular the thinking used during the process that results in reasoning errors may provide insight into the creation of errors and thus provide information for educating end-users to avoid these errors. Hendry and Green (1994) investigated how experienced end-users think while developing and explaining their spreadsheets. The learning of computer skills has also been studied with error-making attributed to trial-and-error approaches (Charney et al., 2001). Kruck, Maher, and Barkhi (2003) found that student logical reasoning increased with spreadsheet training and that their error rates in spreadsheet development decreased. However, they did not extend their work to investigate the thinking behind these errors. Additionally,

Anderson (1989) reported that students made errors using analogical reasoning in arithmetic and algebra. As part of the process in developing a formula in a cell is to represent the situation or problem mathematically, errors are often a byproduct of the problem solving process. These errors are considered as *mathematical reasoning* errors in the taxonomy developed by Rajalingham et al. (2001). Analogical reasoning may play a role in students' progress in learning to develop spreadsheet solutions and the errors they create. Other types of reasoning may be as influential, but little if any research has been done in this area.

The overarching problem for this study is an investigation of students' reasoning when creating spreadsheet applications, to see how that reasoning develops and its impact in the creation of errors in spreadsheet solutions. More specifically, the questions for this study are:

1. What is student's reasoning that results in *domain knowledge* errors in spreadsheet solutions?
2. What is student's reasoning that results in *implementation* errors in spreadsheet solutions?
3. What learning experiences shape students' reasoning strategies that result in either correct or incorrect solution to problems?

Significance of the Study

The decisions made based on spreadsheet solutions require reliable, accurate spreadsheet applications. Yet, few spreadsheet developers have received formal training

in developing dependable and accurate spreadsheets. More and more, development of spreadsheets is delegated to end-users who typically have even less training because of the perceived ease of using spreadsheets for solving problems. Furthermore, the resulting products are often amended to solve additional problems even though the spreadsheets may or may not be dependable and reliable when making changes.

Thus, the issue of reasoning and actions leading to errors in spreadsheets is important as more and more end-users are developing spreadsheets. Methods for guiding these end-users in reducing errors are important and a current topic of research. Most efforts to improve spreadsheet accuracy have focused on detecting errors after they are made. Spreadsheet errors classified as "reasoning errors" often generate incorrect values and may be the main source of inaccurate spreadsheet applications that are used often in making critical decisions. The reasoning involved in spreadsheet development and error creation needs to be better understood if this problem is to be resolved.

Determining the reasoning used during spreadsheet development and how that reasoning is learned will help in the identification of instructional methods directed at reducing errors in spreadsheet development. Identifying reasoning methods that are effective in developing accurate spreadsheets will help shape more effective reasoning, which may reduce error-making and increase error-correction during spreadsheet development. Studying the problem with students in a formal learning environment allows for an investigation of the effects of learning activities and their impact in shaping students' reasoning strategies for solving problems with spreadsheets. Recognition of the impact of learning experiences in association with errors will help identify effective

reasoning strategies and those to avoid. It may be possible to apply the findings to spreadsheet instructional and learning processes could ultimately lead to the development of more accurate spreadsheet applications.

Chapter II

Review of the Research

Framework

End-user programming with spreadsheets is the most predominant method of computer application development, surpassing those written using traditional programming techniques (Hendry & Green, 1994). These spreadsheet-based applications are used to make critical decisions in business, planning, and research. The cost of errors can be significant and has prompted increased interest in methods to reduce errors. In software development using traditional programming languages, a number of techniques have been integrated into the development process to ensure accuracy. Design and testing are prominent components of the software development cycle. In contrast, spreadsheet applications are commonly developed in an ad-hoc fashion, often by an individual. Little or no effort goes into spreadsheet design or testing (Panko, 2000). Therefore, most operational spreadsheets used for routine computations in business and industry contain errors.

Due partly to the business decisions made based on spreadsheets and the economic impact of errors, accuracy is a current interest area in spreadsheet research. Efforts to find and reduce errors include spreadsheet testing, design techniques, software enhancements, and classification. Little has been done, however, on spreadsheet error prevention (Kruck, Mayer, & Barkhi, 2001; Burnett et al., 2003), where either the error is not made or is corrected during implementation. In order to improve spreadsheet error-

prevention, it is necessary to understand how errors occur, what type of reasoning is used that results in errors, and how that reasoning was acquired or learned.

As described in Chapter 1, Rajalingham, Chadwick, and Knight (2001) developed a taxonomy of spreadsheet errors to help explain the nature and impact of the errors. The classification of the taxonomy of interest is *reasoning* errors. This classification includes those errors that are created by entering incorrect formulas, generating incorrect values. These errors are caused by either choosing the wrong algorithm or creating the wrong formulas for implementation of the algorithm planned in the spreadsheet solution. Reasoning errors have been investigated in related fields such as mathematical ; s (Lithner, 2000; Kintsch, 1998; Stevens and Palocsay, 2003) and analogical reasoning (Anderson, 1989).

To investigate how people are thinking when making reasaoning errors, it is necessary to understand how they are thinking while developing spreadsheets. Integrating this information with research on how spreadsheets are learned provides direction for studying how people learn reasoning that leads to spreadsheet errors. The relevant research on spreadsheet errors also needs to be included to provide background on the nature of spreadsheet errors, spreadsheet error measurement, spreadsheet conceptual issues associated with reasoning, and current efforts to reduce spreadsheet errors. The components of this research review (Figure 2.1) are spreadsheet *errors*, *reasoning* used in spreadsheet development, and *learning* spreadsheet skills and concepts. Further background in reasoning and errors is provided in the section on Cognitive

Theory. Finally, the research methods used to investigate these phenomenon are reviewed.

Spreadsheet Errors

In order to understand how students' reasoning causes spreadsheet errors, some background on spreadsheet errors is necessary. This section reviews representative research on spreadsheet errors including the following: classification of errors, user attributes, spreadsheet characteristics, testing, and preventative measures.

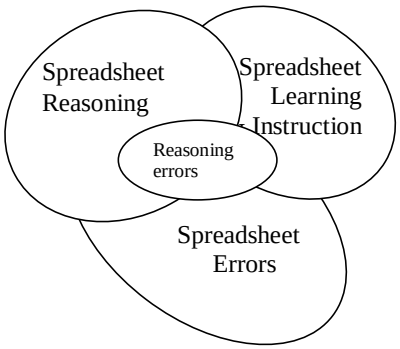


Figure 2.1. Components of research literature for spreadsheet reasoning errors.

Panko's (1998) review of the research on spreadsheet errors revealed that errors occur in a large number of spreadsheets (40-60%). In those spreadsheets, errors occurred in 3-5% of the cells. Panko reported on what was currently known about spreadsheet errors and how errors are measured. Table 1 presents Panko's summary of spreadsheet development experiments. A more detailed analysis of the spreadsheets and errors

indicated that the error rate may be much higher (Panko, 2005). Common errors include references to the wrong cells, cell reference errors caused by copying, misplacement of data in design, mistyping of formula operators, error in command sequence, confusion over formula and values, and reversal of terms in subtraction (Janvrin & Morrison, 2000; Brown & Gould, 1987).

Table 1.

Summary of Spreadsheet Development Experiments

Study	**Sample**	**Study**	**Cell Error Rate (CER)**	**Pct. of Models with Errors**
Development Experiments		*Errors counted at end of development process*		
Brown & Gould [1987]	9 highly experienced spreadsheet developers	Developed 3 models apiece. All made an error in at least one model. Minimum definition of errors, excluding the omission of requirements.		44%
Brown & Gould [1987]		Broader definition of errors including the omission of requirements.		63%
Hassinen [1988]	92 novice spreadsheet students developed 355 spreadsheets	Paper and pencil exercise. Subjects filled in formulas in a skeleton model containing organization and numbers.	4.3% (2)	55%
Hassinen [1988]	10 novice students developing 48 spreadsheets	Computer exercise for same task.		48%
Janvrin &	61 upper	Study 1: Developed model with	7%-14%	84%-

Morrison [1996, 2000]	division business and graduate accounting students	multiple worksheets. Had template with filled-in values. Measured incorrect links between worksheets. Model had 51 links. Students had 16 days to do task. Worked an average of 8.8 hours	(3)	95%
Janvrin & Morrison [1996, 2000]	88 senior-level accounting students	Study 2: Developed model with multiple worksheets. 66 links between worksheets. No templates to work from; only 1 check figure. CER is percent of incorrect links between spreadsheets.	8%-17% (3)	
Panko & Halverson [1997]	35 undergraduate business students working alone	Developed pro forma income statement based on the Galumpke task.	5.4%	80%
Panko & Halverson [1997]	40 undergraduate business students working in groups of 4	Developed pro forma income statement based on the Galumpke task	2.0%	60%

Note: From "What We Know About Spreadsheet Errors" by R. Panko, 2000, *Journal of End User Computing*, Volume 10, No 2., pp. 15-21

Different values are used to assess errors in spreadsheets: the percent of spreadsheets that contain at least one error, how many errors an average spreadsheet has, and cascading errors (those resulting from errors in other cells). Cell Error Rate (CER) is a common measure referring to the number of errors divided by the combined number of numerical and formula cells. The CER was found to be similar among selections of completed worksheets, even though developers ranged from novice to expert. The spreadsheets in the research reviewed by Panko included field studies of business spreadsheets, lab studies with simple spreadsheets, experienced spreadsheet developers,

and novices. Even when a task was simple with no domain specific knowledge required, about 40% of all spreadsheets contained errors with a CER of about 2%.

Several approaches have been proposed for reducing spreadsheet errors. They include development methods similar to software development, domain-specific development tools, and code inspection tools. Spreadsheet programming has become a predominant method of application development, but strict development guidelines that are utilized to eliminate errors in traditional programming are not integrated into the process. Thus far, a similar set of guidelines have not been used for spreadsheets. Tools for spreadsheet design, debugging, and testing have been developed. Many of these tools improve spreadsheet accuracy, but take away from the ad-hoc features of spreadsheet development. In addition, visual cues can be used to illustrate cells in a series and icons to indicate inconsistencies in a formula series.

One method of examining spreadsheet errors is using a classification system to categorize them. The classification of errors may enable researchers to determine which types of errors occur most commonly. Also, a classification system may help users to gain a better understanding of the errors that can occur in spreadsheet applications.

Spreadsheet errors are commonly classified as quantitative or qualitative (Panko & Halverson, 1997; Rajalingham, Chadwick, & Knight, 2001). Quantitative errors are those that result in a wrong value, while qualitative errors are those that affect the accuracy of the spreadsheet after modification or interpretation of the spreadsheet or future updates. In the classification system developed by Panko & Halverson (1997), quantitative errors are further categorized as mechanical (mistyping), logical, and

understanding. Common errors include mistyping a number, entering an incorrect operation (+ vs. -), and pointing to incorrect cells. Panko & Halverson (1997) suggested tools, techniques, or methods to prevent the occurrence of errors or to enhance the chances of detecting the errors. In the process of classifying an error to fit into the taxonomy, developers and end-users might gain a deeper understanding of the error.

The classification system above is useful for analyzing errors in existing spreadsheets, but does not give enough detail to help determine causes of errors adequately. Rajalingham et al. (2001) developed a hierarchical classification of spreadsheet errors based on the nature and characteristics of the error and the spreadsheet development life cycle (Figure 1.1).

System errors are errors made by the spreadsheet software, operating system, or hardware. Accidental errors are mistakes or slips, such as typing errors. They often are corrected and spotted quickly after being made. Omission, alteration, and duplication errors can be made by both developers and users. A key factor or variable left out of the spreadsheet model is an omission error. Alteration errors are due to someone accidentally making a change to an existing model that produces a defect in the model. If part of the model is accidentally recreated, a duplication error is generated.

Errors in reasoning, the focus of this study, involve entering an inaccurate formula due to a mistake in reasoning. Subcategories of *reasoning* errors are *domain knowledge* and *implementation* errors. *Domain knowledge* errors are due to a lack of understanding of the problem necessary to model the solution accurately. Included in this classification are errors resulting from selection of an incorrect algorithm (*real-world knowledge*) and

inaccurate construction of a formula to implement a correctly chosen algorithm (*mathematical representation*). *Implementation* errors are due to lack of knowledge of the full use of the functions and capabilities of the spreadsheet package, with a lack of understanding of the spreadsheet principles, concepts, constructs, reserved words, and syntax. *Implementation* errors are further divided into *syntax* errors and *logic* errors. *Syntax* errors occur when the formula contains characters and symbols that are not recognized by the spreadsheet for performing the desired function. Most spreadsheets have the capabilities for signaling these errors. A *logic* error produces incorrect values resulting from an incorrect construction of a formula; this error may be due to a lack of understanding of the specific features and functions of the spreadsheet, thus producing an incorrect value. Examples of reasoning errors are shown in Table 2, based on the spreadsheet sample in Figure 2.2

The goal of the taxonomy proposed by Rajalingham et al. (2001) is to classify spreadsheet errors with little or no overlap between categories. However, in the spreadsheet segment shown in Figure 2.2, the situation in cell E7 is an example of an error that might be classified in more than one category. The algorithm is incorrect, but could have resulted from a copy operation from the cell above. The situation could be resolved somewhat if one had the opportunity to observe the development process and ask questions.

1	A	B	C	D	E
2	**Models**	**Number Sold**	**Price**	**Income**	**% of Total**
3	**extreme**	13	1438	=13*1438	=D3/D7
4	**Ultra**	34	923	=B4+C4	=D4/D8
5	**medium**	23	750	=B5*C5	=D5/D9
6	**cheap**	13	453	=B6*C6	=D7/D6
7	**Total**	=SUM(B3:B6)	=SUM(C3:C6)	=B7*C7	
8					
9		**APR**	Years	Principal	Monthly Payment
10	Mortgage	**6%**	30	200000	=D10*D10/(12*C10)

Figure 2.2 Sample spreadsheet

Table 2 Reasoning Error Examples by Category

Error	**Example**	**Cell**	**Incorrect**	**Correct**
Real-world (wrong algorithm)	Income	D4	=B4+C4	=B4 * C4
	Monthly payment	E10	=APR*principle/#pmts =B10*D10/(12*C10)	=PMT(APR/12, years*12, Principle)
	Total	D7	=B7*C7	=sum(D3:D6)
Mathematical	Percent	E6	=D7/D6	=D6/D7
Logic	Relative address vs. Absolute	E4	=D4/D8	=D4/D7
Maintainability	Hard-coding	D3	=13*1438	=B3*C3

Rajalingham et al. (2001) described *qualitative errors* as those that do not immediately produce incorrect numeric values but degrade the quality of the model. The model also becomes more prone to misinterpretation on the part of the user. As a result, it also becomes more difficult to update and maintain the model. Qualitative errors can generally be divided into two different types, namely, *semantic* and *maintainability errors*. Semantic errors are qualitative errors that occur due to a distortion of or ambiguity in the meaning of data. This situation can consequently lead to incorrect

decisions, choices or assumptions. Semantic errors are relatively difficult to detect. Subcategories of semantic errors are *structural* and *temporal* errors. Structural errors usually take the form of flaws in the design or layout of the model, and incorrect or ambiguous headings. They include situations in which the documented assumptions are not reflected in the model, causing confusion. Temporal errors are described as qualitative errors produced by the use of data which has not been updated. They can lead to unreliable decisions or interpretations of the situation. For example, if a monthly payment calculation is based on an Annual Percentage Rate (APR) and that value has not been updated, the result will appear to be correct but will not be for the current APR.

Maintainability flaws are features of the spreadsheet model that make it difficult to update or modify the spreadsheet. They can potentially cause inconsistencies in the model. A common and typical example of a maintainability error is hard-coding.

User and Spreadsheet Attributes

Beyond the classifications of spreadsheet errors, errors attributed to user attributes and spreadsheet attributes are also of interest. Panko and Sprague (1999), Panko (2000), Howe and Simkin (2006), Tukiainen (2000) investigated attributes of the student and spreadsheets that may contribute to errors. Panko and Sprague (1999) investigated error rates for different types of subjects in a simple domain-free problem. Errors generated in two simple spreadsheet applications were analyzed by Panko (2000). Howe and Simkin (2006) investigated the possible relationships between student demographics and

background and the types of spreadsheet errors made. Tukiainen (2000) investigated spreadsheet errors in conjunction with the conceptual level of spreadsheets.

Panko and Sprague (1999) and Panko (2000) investigated errors made during development of relatively simple spreadsheets. The same spreadsheet problem was used in both studies and a second spreadsheet was used for Panko (2000). Panko and Sprague (1999) investigated compared spreadsheets and errors created by Masters of Business Administration (MBA) students and undergraduate students. The MBA students were also grouped based on prior spreadsheet experience. The experimental hypotheses were that MBA students make fewer errors in creating spreadsheets than undergraduates and that MBA students with spreadsheet experience make fewer errors than MBA students with little or no spreadsheet experience. A questionnaire was included to collect information about students' background. A subset of the undergraduates were trained in code-checking and given the opportunity to fix errors.

Panko (2000) developed two corpuses of spreadsheet errors. The goal was to provide spreadsheets for assessment of spreadsheet error reduction tools. Two relatively simple problems were developed and given to different groups of students. One problem was the same used in Panko and Sprague (1999)

The results of the first study indicated that the MBAs made as many errors (25%) and similar types of errors as undergraduate students. Thirty-five percent of all spreadsheets had at least one error and the cell error rate (CER) was 2%. Over half (54%) of the errors were omission (5 distinct errors), with the majority of the remaining errors considered as logic errors (16 distinct logical errors). Similar results were

observed in the study by Panko (2000). A notable result from both studies was the small number of mechanical errors. Most errors of this type were transposition errors or the results of misreading a number.

Omission errors were present in the spreadsheets developed in both studies. The researchers felt that the high percent of omission errors may be attributed to the task being larger than students' working memories, which must hold planning information as well as data. In Panko (2000) the more complex formulas in the spreadsheets seemed prone to omission errors. The researcher proposed that if the amount of information was larger than short-term memory could handle a "blend" error might result, containing elements of competing statements. This kind of error would be especially prevalent if the student was engaged in a cognitively complex task like spreadsheet development, which requires retaining a design plan in mind.

The number of distinct logic errors was interpreted to indicate that there may be a strong random element in error-making. In Panko (2000) only a few errors occurred more than three times and many errors only occurred fewer than three times. This means that there would not be a main error or two to look for that would result in major reductions in error-rate. The implication of the types and wide variety of errors found is that auditing a spreadsheet requires a thorough inspection of cells. Error making appears to be extremely diverse. Students may not make the same error in two different spreadsheets.

One error that was common was in conversion between units, for example, when an annual figure needs to be converted to a monthly one. Reason (1990) refers to this as

a “Strong but wrong” error, where the automatic cognitive subsystem saves work by working through pattern matching

One concern with previous spreadsheet experiments had been that the task required too much domain knowledge. But the results of this experiment indicate that even with a simple, relatively domain-free task, subjects still made errors. There was not a large difference between the errors in spreadsheets developed by people with spreadsheet experience and novices in terms of the errors made and types of errors.

The researchers noted the diversity in errors would make it difficult to developing automated software tools to find the errors, as formulas had correct spreadsheet syntax and produced values, but there were flaws in the formula as it related to the problem. A number of errors were omission errors, where part of the problem or model was left out. Examining formula structure would not help find these errors so a software tool would not be an effective method of reducing these errors.

To determine who makes errors, the subjects’ self-ratings in spreadsheet ability from the questionnaire were compared with their error rates. No notable differences were found between students who make errors and those who do not. The author suggested that a questionnaire may not be an effective instrument for this kind of analysis.

One of the areas that has been investigated is the debugging of spreadsheets, finding errors. In Panko and Sprague (1999), to determine how well students would do finding errors in their own spreadsheets, a subset of the undergraduates were given 10 minutes of in-class instruction on spreadsheet errors and another 10 minutes of instruction on how to code-inspect a model spreadsheet. They were shown how to check

facts in the model against those in the problem statement, and they were introduced to Excel's auditing tools for cell referencing. These subjects were then asked to code-inspect their spreadsheet. Only 13% (3 of 23) of the subjects who had errors in their spreadsheets were able to make appropriate corrections. A total of 21% of their errors were fixed, but one new error was introduced in the process. This rate is considerably less than the 50% found in Galletta, Hartzell, Johnson, Joseph, Rustagi. (1997). The authors attribute this lack of accuracy to the students' optimism in inspecting their own models.

Panko and Sprague (1999) analyzed the effectiveness of one small set of undergraduate students finding errors. Panko & Halverson (1997) analyzed the effects of students working in groups and found that although the error detection rate improved, there was an upper bound on the effectiveness of groupwork. Howe and Simkin (2006) designed a study to determine whether or not factors such as age, gender, year in college, experience with spreadsheets, academic ability, and confidence affected the ability to detect and correct various types of spreadsheet errors. A goal was to identify factors which might help predict an individual's ability to detect spreadsheet errors. The research hypotheses were as follows: that factors such as age, academic ability, and spreadsheet background positively affect spreadsheet error-detection; that gender would impact error-detection; that error-detection rates would be similar among various types of spreadsheet errors. The study used a pre-test questionnaire, an Excel spreadsheet, and a post-test questionnaire to measure demographic factors and ability to detect spreadsheet errors.

Students (n=228) from two different colleges volunteered for the study. These students represented a variety of majors, ages, and backgrounds, and were equally distributed between genders. Academic ability was estimated through grade point average (GPA). A spreadsheet was used comprised of three worksheets that contained documentation, payroll calculations, office expenses, and projections. The spreadsheet had been seeded with 43 errors of various types: Clerical, Data Entry, Formula, and Rules violation. Qualitative errors included items mislabeled, misspelled, and incorrect dates whereas the sources of most quantitative errors were inaccurate data, formulas, and cell references generated quantitative errors. The researchers also included "Rules Violations" which violate company policies for particular variables. For instance, only eligible employees can work overtime.

The participants, on average, found 67% of the errors in the spreadsheet. This result is similar to those in previous studies in which participants found approximately one-half of seeded errors (Galletta et al., 1997). The results indicated that there was little or no impact on spreadsheet error-detection due to any of the factors. Error-detection was also uniform among the various types of errors. The authors suggest that none of these factors, including a person's self-confidence in his or her ability, can be used to predict their ability in spreadsheet error-detection.

The researchers noted several limitations to their study which are common to most of the lab-based studies in the literature. A student sample with a small spreadsheet is not representative of industry spreadsheet users and of developers' experiences. The factors studied were self-reported and may not accurately reflect the persons'

backgrounds. Length of time people have used a spreadsheet does not reflect on the quality of their experience.

Tukiainen (2000) noted that the low-level conceptual level of the spreadsheet is often cited as a source for errors. Several aspects of spreadsheet layout leave the developer with a limited view of the true spreadsheet structure: formulas are not visible all the time, visibility is limited at any one time to the size of the display, naming cells is the only way to structure computations, specification of absolute and relative referencing is vague.

BASSET, a spreadsheet program with structuring capabilities, was used to address some of the noted problems. BASSET has the capabilities to make structures explicit so that users can refer to them rather than to specified sets of cells. The structures are connected to computation, and changes in structures are reflected automatically. This experiment by Tukiainen compared the results of two groups of novice users' learning of structured spreadsheet development. One group used BASSET while the other used Microsoft Excel. All participants then worked on four tasks composed of typical spreadsheet construction and modification operations: data entry, composing formulas with relative and absolute addressing, copying formulas, and saving.

Errors were categorized into broad classifications of spreadsheet computation (errors due to the user's mental model of spreadsheet-based computation and mathematical concepts), calculation paradigm (errors attributed either to Excel or BASSET), and tool usability. The two groups generated different types of errors. Some errors were due to BASSET requiring more initial effort in choosing goals and planning

compared to Excel allowing a user to work locally and at a low level. Since BASSET is based on establishment of data areas with labels, more work is required with multiple windows and menus. Spreadsheets developed using BASSET had fewer errors, particularly when an accumulation over a structure (i.e. summation over a series) was required. Inserting new rows and columns into existing data sets created problems for Excel users, while BASSET incorporated the new data into the existing data structure correctly for users.

There were fewer BASSET-related errors than Excel related errors. The authors suggested that the higher conceptual level of BASSET helped novice users with more complex tasks. BASSET may require more time in initial training to introduce the concept of goals as a system feature.

This study noted some good examples of novice type errors and interpretations of spreadsheet paradigm:

- misunderstanding of spreadsheet concept – difference between constant value and reference to a cell containing that constant value
- fragmentary knowledge of spreadsheet operations – the side effects in inserting an area into a spreadsheet
- inability to devise a solution for a cognitively difficult problem
- errors in mathematical models underlying the spreadsheet implementation of formulas.

One of the difficulties noted was that the higher conceptual level of BASSET forced novices to think globally, increasing the mental load. Novices tend to think locally, and the discrepancy was evident in tasks comprised of a chain of goals, where the

novices may not have been able to transform the problem adequately into subgoals in BASSET. Novices also have to master the goals available in BASSET to know what computations are available in the system. The researchers suggested that perhaps more time was needed to learn about BASSET. More time may offset some of the benefits of learning to develop applications using spreadsheets, but the study may also indicate a direction that is necessary, similar to modular programming with traditional languages.

Design and Prevention

There have been few efforts towards preventing errors in spreadsheets including instructional techniques, design methods for reduction of referencing errors, and building safeguards into the spreadsheet. Peer learning techniques and auditing methods used in industry were integrated into student learning by Chadwick and Sue (2001). Janvrin and Morrison (2000) and Whittle and Janvrin (2002) introduced methods to improve referencing accuracy. Wallace, Cook, Summet, & Burnett (2002) and Abrahams & Erwig (2006) added capability for separation of design and implementation, as well as thi capability to develop standard templates and correct update operations.

Chadwick and Sue (2001) conducted an experiment that blended educational methods with industry methods. Self-assessment has been used in education to enable students to reflect on the quality of their work. Peer assessment is also used in education to give students appreciation for their work and experience in identifying common mistakes. Self-audit techniques in this study were based on industry auditing methods. The researchers proposed that use of self-assessment and peer assessment techniques

educated students in a set of structured, transferable skills for spreadsheet error-checking. Familiarizing novices with others' mistakes and misconceptions might reduce the number of errors. Introducing risks associated with incorrect models might also encourage students to apply error-checking controls during development. Students are generally taught how to do things, and perhaps instruction should include how not to make errors.

Lecture sessions were designed to improve awareness of common errors, causes, and effects and encouraged students to consider the benefits and costs of spreadsheet audit. Students were also encouraged to reflect on the need to find errors, the usefulness of auditing, and the benefits of self/peer assessment for spreadsheet models. The participants were second year undergraduate information systems students. They were considered "fairly novice," as they had little spreadsheet background and had not yet formed bad habits that might lead to poor spreadsheet design. Students were given written specifications for a spreadsheet to develop a working model. The worksheet was then audited by the developer and peer-reviewed by another student in the class. Students also audited spreadsheet models seeded with errors.

A questionnaire was used to assess the project and the usefulness of techniques employed. The results were favorable in terms of students feeling that peer review and audit procedures might be beneficial. The comments indicated additional motivation to do well if their work was to be viewed by others. It was not reported whether error rates were affected by the lectures, self-audit and peer assessment.

Two studies investigated methods to improve referencing errors. Janvrin and Morrison (2000) studied the impact of a structured design approach on spreadsheet error

rates and development time and Whittle and Janvrin (2002) investigated the effect of using cell names (range names) on error rate.

In the first study, Janvrin and Morrison (2000) investigated whether use of Data Flow Diagrams (DFD) in design could reduce errors in referencing between worksheets. One way of creating some modularization with spreadsheets is to create separate worksheets for various components of the model and to reference associated cells.

In this experiment, the participants received one hour of spreadsheet training and then were assigned a problem containing 10 separate worksheets with 51 referenced values among the worksheets. The participants in the experimental group were required to submit a design of the spreadsheet prior to beginning. Sample values were given to provide check figures and to reduce other errors. A pre-test questionnaire was used to record self-reported pre-test spreadsheet expertise. A post-questionnaire was used to measure spreadsheet expertise and to design a confidence coding.

Spreadsheets developed by subjects who used structured design techniques had a significantly lower rate of error than those who did not. The authors believe the design methodology forced developers to identify references between worksheets before they were implemented. Almost all referencing errors were due to hard-coded numbers, rather than to an incorrect reference. Pre-test spreadsheet experience was a significant predictor of post-test expertise and reduced error rates. Subjects with higher pre-test Excel knowledge made fewer conceptual errors and overall errors. Domain (accounting) experience was a positive indicator of post-test spreadsheet expertise and design condition. As the problem was complex, it was reasonable to assume the domain

expertise would contribute.

Students working in pairs had fewer errors, regardless of which group they were in Panko and Halverson (1997) found that groups of students discovered more errors than people working individually. Perhaps the groups in this experiment were able to find more errors and correct them during the development phase.

Common errors made by students included their use of values instead of cell references (hard-coding), incorrect cell references, and non-uniform cell references (Janvrin & Morrison, 2000). Although hard-coding is considered a qualitative error, an argument can be made that it is a fault in reasoning (Clermont, 2003) as it reflects lack of understanding of spreadsheet capabilities. Hard-coding errors are also significant, as they can generate incorrect values during updating and maintenance.

The authors concluded that domain and spreadsheet expertise impact error rates. The structured design approach specifically identified references during the design phase and resulted in fewer referencing errors. Other methods of identifying potential error area before implementation may also improve end-user application reliability.

In the study by Whittle and Janvrin (2002), the main hypothesis was that use of cell names would improve spreadsheet accuracy by reducing errors, and that the use of cell names would increase concept retention. The study involved a task in which participants were to complete a spreadsheet template for a capital budget decision. The subjects were familiar with the domain and spreadsheet skills necessary. The template was composed of four worksheets: input, two different calculation methods, and comparison of methods. Specific instructions were provided, including a set of values to

check results. The treatment group received one hour of instruction in naming cells.

Results indicate that naming cell ranges lowered calculation errors and referencing errors significantly and adding cell labels (range names) did not impact development times. The percentage of error-free spreadsheets generated by the experimental group was less than the control group ($p<.10$). This research showed that spreadsheet accuracy can be improved through use of cell labels (range names) and that cost of development (in terms of time) is not increased. The authors also began a preliminary analysis to determine how use of cell names influenced spreadsheet maintenance and auditing.

Although the subjects in this experiment had an existing background in spreadsheets, the results might also be applicable to spreadsheet novices. Range naming is often included in novice learning but is not emphasized. This research would indicate that range naming may lower errors for novices. The previous study dealt with improving cell references between worksheets using design diagrams. As noted previously, design is not generally part of spreadsheet curriculum although it is an integral part of programming curriculum. Even though novices generally begin working with a single spreadsheet, they often begin to use more than one sheet in order to achieve some degree of modularity. Learning to design references between sheets could be part of the overall spreadsheet instruction.

The previous studies dealt with instructional techniques and design methods. Another approach to reducing spreadsheet errors is to build safeguards and cell criteria into the spreadsheet. Wallace et al. (2002) developed a method of integrating conditions

for cells into the spreadsheet during development. A methodology involving templates and update operations was developed by Abraham and Erwig (2006).

Wallace et al. (2002) used a think-aloud study to investigate students' reasoning in using assertions in spreadsheet maintenance. Assertions allow users to communicate aspects of their mental model of the problem beyond what is in the formulas and to provide a method for programmers to reason about the integrity of their logic and document assumptions, and to find errors. Assertions take the form of preconditions, post-conditions, and invariants. In the study, college students were provided with training and worked on spreadsheet applications. A control group had no assertions; the treatment group did. Two spreadsheet modification tasks were used as activities. The study data included audio recordings of the subjects thinking aloud and electronic transcripts of their keystrokes. Researchers' observations of subjects during the spreadsheet activity and post session questions gave additional information.

The students who used assertions were able to find more errors during the modification tasks than the control group. Several behaviors were observed which were unexpected. Subjects tended to use a "local testing" strategy. They tested only the information in a single cell, for example, checking math. Subjects also avoided testing using "easy" inputs, which would have allowed them to use their domain knowledge to predict and easily determine the accuracy of their answers. The students viewed these inputs as unrealistic test values or felt that the computer could do them by default.

Abraham and Erwig (2006) developed a system with which a developer can generate a template for spreadsheets and a set of correct update operations. A key

element of their approach is separate of modeling/design phase and implementation/data entry phases of development. A software modeling tool was developed that allows developers to create templates for spreadsheets and update operations. This approach can prevent certain kinds of errors such as range, reference, and type errors. The researchers have extended their work so that a template and update operations can be created from an existing spreadsheet.

Results indicated that the extraction system developed better templates than either novices of experts while the expert group created templates that were significantly more correct than the novices. Video-recordings were made of the subjects' interaction with the spreadsheets. The amount of time and number of cells inspected to infer a template were measured. A study designed to investigate the effectiveness of this method demonstrated that the system considered performs significantly better than subjects with intermediate to expert level programming expertise

Reasoning in Spreadsheet Development

The reasoning people use during spreadsheet development has been investigated to some extent, as reported in the literature. This includes research on thinking and communication used during spreadsheet application development, mental and cognitive models, and types of reasoning used. Observations, interviews, experiments, standardized assessments, and data recording have been used to study these phenomena.

Nardi and Miller (1991), Hendry and Green (1994), and Brown and Gould (1987) conducted classic studies to investigate how people develop spreadsheet applications and

how they think during development. Nardi and Miller (1991) performed an ethnographic study of spreadsheet users as developers in the workplace, reporting that spreadsheets were used as cognitive artifacts, as a way of sharing information. The results indicate that the development effort was highly collaborative. In contrast to traditional programming, where a problem is given to an expert to design and develop a program, in spreadsheet development, it is often the end user, as the domain expert, who is responsible for development. Unsophisticated users can write programs in the form of formulas that establish numerical relation between data values. Users transfer domain knowledge via spreadsheet templates and direct editing of spreadsheets. Workers with different skills and levels of expertise were found to work together to verify correctness of spreadsheet applications. Users gradually learn new techniques from other users. The study touches on spreadsheet errors, noting that most developers check for reasonableness of answers.

Hendry and Green (1994) employed an interview methodology similar to Nardi (1993) to study spreadsheet development process of individual users. Interviews were conducted to determine what was easy and hard about spreadsheets, how developers checked for errors, how they went about understanding a spreadsheet, and how they explained a spreadsheet. The spreadsheets used varied in size, complexity, importance, and time to create.

Several difficulties were found for creation of spreadsheets:

- If the spreadsheets formulas closely match the requirements of the problem domain, creating cell formulas was easy. Some problems appeared to be simple but could not be solved easily and required combinations of functions. The solution may require specialized

spreadsheet functions that may have no place in the user's knowledge of how to solve the problem within the domain of work.

- There are several techniques for cell referencing, which when combined with copying techniques give users many options for implementing computation structures. The participants indicated that a common source of error was using the incorrect referencing technique. One experienced user recalled having problems with copying functions early on, and then learned to exercise care and use methods that reduced errors. Vigilance in checking references carefully was suggested by another user.

- The participants indicated that they had trouble spotting mistakes and debugging formulas, so they employed design strategies that aid in error checking and debugging. They planned for mistakes and used specific techniques for spotting mistakes, such as systematic testing of formula chains; using carefully prepared test data; computing results by using two different methods; breaking large formula into smaller chunks, using consistent design practices so similar debugging techniques can be used.

Part of the study also involved explaining existing spreadsheets. Results indicated that the cognitive overhead of scrolling and associating cells to data headings complicates goal management in comprehension of spreadsheets. Also, formula in adjacent rows or columns may look similar, which can make it difficult to keep them straight. When making modifications to formula, the authors expect the demands on working memory to be higher than for formula comprehension. Scrolling across a large two-dimensional surface, scanning for headings, trying to associate cell references with domain interpretations, recalling which subtasks have been done, and remembering which still need to be done severely taxes working memory.

The authors assessed the strengths and weaknesses of spreadsheets using the idea of Cognitive Dimensions, which categorizes elements of programming languages. They concluded that, even for simple problems, spreadsheet formulas are not always easy to

create or understand.

Brown and Gould (1987) performed a study designed to find out how experienced users develop spreadsheets. The researchers observed and timed the development process and recorded keystrokes. One focus of the study was user-generated errors. Participants were experienced spreadsheet users, many of whom had programming experience. Although participants were confident that the spreadsheets they developed were accurate, 44% of the spreadsheets contained user-generated errors. Most of the errors were formula errors with a small percentage of errors caused by mistyping. Causes of formula errors included references to the wrong cells, cell reference errors caused by copying, misplacement of data in design, mistyping of formula operators, error in command sequence, confusion over formula and values, and reversal of terms in subtraction. The researchers also recorded "redoing" of data and formulas, presumably fixing errors. There were a small number of these "redoing" events, and they represented a small percent of time spent.

In development, the general trend was to first enter headings, then numbers, then copy values and formulas. Nearly all copying operations took place immediately after the corresponding data entry or composing episodes. Not much time was spent on design or planning and debugging by these experienced spreadsheet users. However, 21% of their development time was spent paused, presumably planning the next phase.

The researchers note some considerations in generalizing the results. The problems solved with the spreadsheets in the experiment were relatively simple. In addition, the problems were more well-defined than might be encountered in the real-

world. The researchers also conjecture that participants may have been less cautious in the study than in the real world. This may be a consideration in a spreadsheet course as well. Students may not take the problem solution or errors as seriously as they would in real-life. Chadwick and Sue (2001) alluded to this attitudinal feature when trying to impress upon students the impact of errors.

Research on the Cognitive and Mental Models of Spreadsheet Development

When users calculate a value in a spreadsheet, they are using the spreadsheet as a *cognitive tool*. Jonassen (1995) defines a cognitive tool as "generalizable computer tool that are intended to engage and facilitate cognitive processing". This tool is both a mental and computational device that supports, guides, and extends the thinking processes of the user. Jonassen proposes that students cannot use these tools without thinking deeply about the content that they are learning, and second, if they choose to use these tools to help them learn, the tools will facilitate the learning process.

Jonassen (1995) describes the cognitive processing taking place when a person is using a spreadsheet. Developing a spreadsheet requires users to to perform abstract reasoning and become rule-makers and. Creating formulas to calculate values from cells requires that users identify patterns and relationships among the data they want to represent. Next, the relationship must be modeled mathematically, using rules to describe the relationships in the model (Jonassen, 1995).

Spreadsheets are now being used in application development that traditionally would have required programming expertise and resources. The cognitive aspects of

spreadsheets are often compared and contrasted to traditional programming languages. Spreadsheets suppress the complexities necessary in programming languages for simple operations such as summation and handling input and output (Hendry & Green, 1994). Nardi (1993) and Hendry and Green (1994) found that end-users described their spreadsheets and development in terms of the domain, rather than in terms of the programming language. In spreadsheet application development, lower level goals deal with application domain data more closely than with procedural programming, making it easier for users to state plans and goals in terms of application (Sajaniemi & Tukiainen, 1996). The strength of spreadsheets is due to familiar and concrete representation of data values and formulas.

Spreadsheets also reduce much of the work necessary with traditional programming, and are conceptually easier to learn and to use than a traditional programming language, with a less complex mental model. Developers do not have to be concerned with sequential control, changes can be made easily, aggregate operations are easy to implement, and the spreadsheet interface provides immediate output display (Sajaniemi & Tukiainen, 1996). Writing a program using a conventional programming language is complicated by the different ways of viewing a program: linear order, control flow, data flow, and modular structure. Spreadsheets are not processed in linear order and have no control flow other than that dictated by dependencies, and these are generally invisible to the user. Spreadsheets have dependencies between cells, which is similar to data flow in traditional programming languages. Modular structure as it is

thought of in traditional languages is not a part of spreadsheets, but cells tend to be grouped by function (Robins, Roundtree, Roundtree, 2003).

Traditional programming languages are based on data and procedure abstraction. Developers need to have a physical and logical view of the program and the relationship between the program and the problem. The abstraction layers in programming languages distance the problem solving environment from the problem domain. The focus may be on the details of program development, with the problem losing significance. There is thus a tendency to develop a general solution to a problem when writing a program. The spreadsheet programming development process, in contrast, does provide a basis and mechanism for developing abstract views of the problem. Spreadsheet development provides facilities for problem solving at a concrete level, supporting finding a specific solution to an instance of a problem (Wong, 2003).

Isakowitz et a.(1995) and Sanjienemi and Tukiainen (1996) proposed logical and physical models for spreadsheet applications. The logical model consists of a formal description of logic and data structures, regardless of implementation. The physical perspective refers to user interface, formatting, and storage.

In the physical model, a spreadsheet model is a two-dimensional grid of addressable cells, each bound to either a constant or a calculated value. This is the user's view of the spreadsheet. An implicit logical view is the set of functional relations between cells. In the Isokowitz et al. (1995) model, function relations had two types of attributes, data and functional. Data attributes are associated with cells having constant values. Functional attributes are bound to functions, which are calculated and

recalculated. The set of functional relations is called a schema. Every spreadsheet can be characterized by the combination of schema + editorial + data + binding. The editorial component includes the column and row labels, and binding refers to the logical-to-physical mapping that binds the schema, editorial component, and data to the spreadsheet grid. In the normal development process the components (schema, editorial, data, and binding) are not distinct. Isakowitz et al. felt this lack of distinction leads to errors in spreadsheets and that it will be difficult to develop systems that promote the construction of consistent and valid spreadsheet models.

Sajaniemi and Tukiainen (1996) described the spreadsheet in terms of logical and physical models in conjunction with goals and plans. Spreadsheet developers need to maintain domain/logical model and spreadsheet /physical model at the same time. Cells must be mentally chunked together for larger structures to facilitate both spreadsheet comprehension and data manipulation. Goals are associated with the domain/logical model, and plans are represented in the physical/spreadsheet model. Spreadsheet users tend to use subgoals that deal with data that has a clear meaning in the application domain.

Sajaniemi and Tukiainen (1996) found that a spreadsheet user usually has little or no knowledge of system design, so it is difficult to find out how spreadsheet planning is done. The planning that is present is like opportunistic planning: the user/programmer can have a plan existing at different levels of abstraction simultaneously and may alternate between levels. However the developers plan, they use cognitive structures for spreadsheet programming knowledge

Related research on cognitive skills and reasoning.

The relevant research on cognitive skills and reasoning that may help provide a basis and method to answer the research questions posed in Chapter I will be summarized here. There is little research directly on cognitive skills used during spreadsheet development. Kruck, Maher, and Barkhi (2003) proposed that during spreadsheet model development, end-users engage in problem solving, planning, and perceptual motor functions. Information is retrieved from long-term memory to design a spreadsheet model for a problem domain. Remembering task structures and information puts a high load on working memory. These researchers developed a study to investigate the effect of spreadsheet training on cognitive skills was investigated by. There has also been some research done on reasoning used during word problem solving with spreadsheets (Wilson, Ainley, & Bills, 2001).

Kruck et al. (2003) analyzed performance in cognitive skills that would be affected by spreadsheet training. For the study, cognitive skills were selected that can be associated with spreadsheet development and were found in the research literature to increase from training: logical reasoning ability, spatial visualization ability, mnemonic skill, sequencing ability.

Logical reasoning is defined as the ability to reason from premise to conclusion, or to evaluate the correctness of a conclusion. Spatial visualization is the ability to manipulate or transform the image of spatial patterns into other arrangements. In spreadsheets, spatial visualization refers to the ability of the end-user to visualize patterns as row and column vectors and to visualize the relationship between these vectors and

other cells. Mnemonic skill is the ability to encode and organize knowledge as one learns to facilitate retrieval. Spreadsheet users must determine which factors are important for a particular situation and organize the material to develop an application for the specific problem. Sequencing is the ability to put a number of individual operations into correct sequential order to solve a problem. The authors propose that spreadsheet training will influence these four cognitive skills and that these cognitive skills will influence the errors in spreadsheet models.

The researchers noted that although some of the cognitive skills had been examined in the framework proposed for this study, few studies had been longitudinal. Changes in cognitive skills are likely to require several weeks or months. The results indicated that there is a significant increase in logical reasoning after a six-week spreadsheet training period. An increase in logical reasoning was associated with effective development of competent spreadsheet models. These findings help understand and define cognitive changes that occur in individuals as they undergo formal spreadsheet training. The researchers suggest that the association between spreadsheet training, logical reasoning, and error rate needs to be studied further.

Wilson, Ainley, and Bills (2004) investigated students' reasoning while using spreadsheets. Think-aloud protocols and semi-structured interviews were used to find features of spreadsheets that might support a learner's generalization for formulas. The Purposeful Algebraic Activity project explored the potential of the spreadsheet as a tool in the introduction of algebra and algebraic thinking. Students were studied working in pairs, verbalizing the process, and were then interviewed. Transcripts of interviews were

annotated to include non-verbal behavior and written work. Data included a videotape of the student interaction and screen recording during lessons. In addition, the teachers' interactions, field notes by author, and examples of written work and spreadsheets were compiled.

Analysis of the data resulted in identifying features of working in a spreadsheet environment that support pupils' generalizational activity: focus on calculations, use of notation, and feedback. Using the spreadsheet, students were able to generalize the solution to problems. The reference problem is calculating how many chairs would be used based on number of tables laid end-to-end. The reasoning would begin with: "two per table and then adding two at the end", be translated to "times two plus two", and then formalized in a spreadsheet: "=A2*2+2'". In a paper and pencil environment, learners read their written generalization in an idiosyncratic way, but in the spreadsheet, the activity of writing the formula necessitates expressing the calculation in a formalized way.

Evidence indicates that students are thinking of variables when constructing a spreadsheet formula rather than thinking of the particular number in the cell, therefore using notation. In addition, it appears the students are fixing errors, as the researchers note that the quick feedback of a spreadsheet encourages student to adjust their formula to achieve success. Judging whether feedback is reasonable and considering what the spreadsheet has done in its calculation can be important to its accuracy.

Types of reasoning

The reasoning used in spreadsheet development and in learning spreadsheet skills might be analogical, case-based reasoning, means-end, trial-and-error, or imitation. Mayer (2002) describes the analogical reasoning process (shown in Figure 1.3) as recognizing a base problem that is similar to the target, extracting the algorithm, and mapping the solution to the new problem.

Case-based reasoning (CBR) is similar to analogical reasoning but assumes that learning is based on characteristics of specific instances from similar situations in the past. Also, CBR does not assume understanding of the base case, although this understanding gives the best results. Some researchers (Robertson, 1997; Lithner, 2000) argue that often, when analogical reasoning is assumed to be in use, the process taking place is actually a form of imitation or mimicry, based on surface features of examples. As research into types of reasoning used in spreadsheets is scarce, relevant research from related areas is included.

Analogical reasoning is often used in learning new skills and solving problems. Students may base their approach to developing an application by making analogies to a similar problem they have dealt with in the past. Errors may develop due to flaws in the analogical reasoning process. Anderson (1989) described errors resulting from analogical reasoning using computer-based intelligent tutoring systems.

The tutoring systems had been developed to teach students problem solving skills in the domains of Lisp programming, geometry, and algebra. Errors were classified as slips, importations of prior misconceptions into a new domain, and within-domain

misconceptions. A slip was characterized by not being consistent, and the subject can self-correct when the error is pointed out. Slips appeared to increase in frequency with working memory load and to decrease with practice. Errors in the second category—importations of prior misconceptions—had been found previously in physics but were not a factor in this study as the subjects were working primarily in one domain and had few prior conceptions. The misconception within-domain was a consequence of the learning that takes place in the domain. This last category was the focus of the discussion, as the author felt that these errors reflected the learning process. Also, these errors required explanation by the subjects, in contrast to slips, which were easily remedied. Errors resulted from students attempts to bridge points in their problem solving where they had inadequate knowledge. The points causing the problem are often called *impasses* and the bridging process produced *repairs*. Permanent misconception resulted if subjects believed their repairs. If subjects invented the repairs for a particular instance, an inconsistent pattern of errors resulted. Anderson (1989) argues that errors resulting from repairs are caused by making analogies to misleading examples of a problem solution.

Anderson (1989) found instances in solving arithmetic and algebra problems where the solution to the current problem is based on solutions to previous problems, but the solution was not completely effective for the current problem. *Einstellung* errors occur when students have a run of success with a particular solution pattern, and they are likely to repeat this process on a problem when the pattern is no longer appropriate. The researcher attributed this kind of error to choosing inappropriate examples. Anderson (1989) notes that there could be disputes over the nature and number of errors categorized

as due to faults in analogical reasoning and that this inconsistency should be investigated further.

There have been efforts in mathematics education research to investigate the components of analogical reasoning using spreadsheets. Sheehan and Tessmer (1997) investigated the relationships between mental models, declarative knowledge, concept learning, and problem solving in students using spreadsheets. Their results indicate that mental models are comprised of conceptual relationships more than of declarative knowledge. Scheiter and Gerjets (2002) investigated the effect of problem sequencing with respect to analogical reasoning. One of the significant observations was that students would often pick the easiest problem as the next to be solved, rather than the most effective problem. Bernardo (2001) studied analogical problem construction and transfer in mathematical problem solving using spreadsheets. Students who could construct problems by analogy effectively did better in transfer since they understood structural elements of spreadsheet, rather than merely surface features

Stevens and Palocsay (2003) developed a method of teaching linear programming using spreadsheets. The underlying idea was to develop a representation of the problem in terms of measurable quantities that could be translated, via application of an explicit rule, into the proper algebraic form and equivalent spreadsheet formulas. They based their approach on related cognitive psychology and mathematical education research on word problem solving. The proposed reasoning process was that when students encountered a familiar problem type they invoke empirical rules based on information retrieved from memory that were specifically useful in setting up and solving that type.

If the student did not recognize a problem type, a general line-to-line translation procedure was used. The researchers suggested an approach is based on translations, providing consistent a step-by-step formulation method that emphasizes understandable concepts over mathematical symbols. The results indicate that this approach discourages many of the most common errors students make.

Robertson (1997) argued that often what is called analogical reasoning may, in fact, be some form of imitation. An example of this type of phenomenon was found in research on calculus word problems. Lithner (2000) performed a series of think-aloud studies to try to determine the main characteristics and background causes of students difficulties in solving mathematical tasks using calculus. A common characteristic of the students was to be more focused on what is familiar and remembered rather than on even elementary mathematical reasoning and accuracy. In situations where a student's work is not a straightforward implementation of a provide solution process, mistakes are made in local solution steps. Students applied simplified strategies using superficial reasoning when solving calculus problems. They based their strategy on identifying similarities and mimicking procedures instead of on understanding fundamental mathematical concepts. This phenomenon has been called "suspension of sense making," since the subjects are ignoring real world aspects of the problem context and are treating problems as puzzles that require the selection and application of some operations.

Case-based reasoning

The reasoning used in spreadsheet development may be case-based reasoning, which is similar to analogical reasoning but emphasizes concrete components of specific

cases. Case-based reasoning (CBR) integrates memory, learning, and reasoning. As a cognitive model, CBR emphasizes the concrete cases, experience, over the traditional theories of cognition general-purpose abstract operators (Kolodner & Guzdial, 1999).

A case solution can be reused on a new case. This involves identifying the differences between the two cases and deciding what part of the retrieved case can be transferred to the new case. A person may focus on similarities between the two and apply the solution of the retrieved case directly to the new case. If differences between the two cases are taken into account, an adaptation process is used on the retrieved solution. Transformational reuse focuses on the equivalence of the solutions while derivational reuse looks at how the problem was solved in the retrieval case. New subgoals can be pursued based on the old ones, and subplans associated with them can be retrieved.

Expert-Novice Reasoning

One of the areas of interest in a study on reasoning, learning, and error-making is the differences between experts and novices. Presumably, experts operate at a different cognitive level than novices and make fewer errors. In most areas, experts and novices use different kinds of reasoning when solving problems. In chess, physics, and computer programming, experts operate with bigger chunks of information. They are able to form cognitive structures, schemas, which can be retrieved. Novices reason using concrete experiences (Bransford, Brown, & Cocking, 2000). A goal in instruction and learning is to impart, in some way, the skills, knowledge, and reasoning of experts to novices using wizards, interactive help, tutorials, and instruction.

In traditional programming languages, expert mental models were found to be based on recognition of basic patterns or schemas which are hierarchical and multi-layered, with explicit mappings between layers and good internal connections, and which are well grounded in the program text. Novice representations generally lacked these characteristics (Robins, 2003). In spreadsheet development, novices focus on localized sections of the spreadsheet, without a global view of the application (Tukianen, 2000)

Hendry and Green (1994) found that when solving unfamiliar problems in some areas, experienced developers gave the impression of being little more than novices. In situations where end-users were faced with a problem but saw no way to approach it, they often concluded the problem was insoluble, even though it might have appear at first to be solvable. In other situations, people either consult a mental model of another abstract, make a high-level interpretation of the situation, or find a command that appears to be associated. The first approach is often ineffective because it is difficult to formulate a model of spreadsheet operation due to the diversity of functions, referencing techniques, etc. The second approach is difficult because many functions have bland names or names that do not give cues suggesting solutions to the problem. The subject appears to be blocked from using either method and may have trouble forming an appropriate plan.

Research on How Users Learn to develop spreadsheets

Spreadsheet errors may be the results of the learning process. The reasoning errors described by Rajalingham et al. (2001) include real-world errors, mathematical errors, and logical errors. Real-world errors are caused by not choosing the correct

algorithm, which could be due to not knowing the correct algorithm exists or due to having learned it incorrectly. The mathematical errors could be due to the learning background in mathematics. Logical errors are caused by improper use of a spreadsheet formula or function. Students do not understand the spreadsheet capabilities, indicating they have not learned it correctly.

Jonassen (1995) described the processes involved in learning to use a spreadsheet. Building spreadsheets requires abstract reasoning by the learner, with the functional and graphical capabilities of the spreadsheet providing a context to engage students in analyzing and connecting multiple representations. Spreadsheets can be used in situations with complex quantitative relationships. The "what-if" thinking supported by spreadsheets requires learners to consider implication of conditions or options, which entails higher order reasoning.

Jonassen (1995) suggests that spreadsheets can be used as a *cognitive tool* to solve problems. When used effectively, this "tool" can enhance performance and facilitate learning. Learners' understanding of algorithms and mathematical content used to describe content domains can be enhanced through identifying values and developing formulas to interrelate values in a spreadsheet. The spreadsheet process models the mathematical logic that is implied by calculations. Making the underlying logic obvious to learners should improve their understanding of the interrelationships and procedures.

Students may have difficulties translating situations or problems into a spreadsheet format. Some of the challenges may be similar to translating word problems into mathematical representations. Many spreadsheet applications are developed based on

a verbal description of a problem or a situation to model. A problem often needs to be translated into an intermediate mathematical representation prior to being implemented into a spreadsheet representation. Kintsch (1998) identifies two steps in solving word problem that give students difficulty: formulation of a mathematical model of the problem and determination whether or not a correct model has been formulated. Algebra word problems have a significant cognitive component that requires construction and manipulation of representations of the problem elements. Students begin to exhibit difficulties with this cognitive process as early as sixth grade, and those difficulties can persist for years.

Rutledge and Rhea (1991) reported on a project designed to introduce spreadsheets into the classroom and curriculum. The changes in curriculum and student use of spreadsheets were studied via observation and interviews. One goal was to determine what cognitive levels were being used by students and teachers during instruction and learning. The upper four levels of Bloom's taxonomy were associated with spreadsheet activities:

a) Application - Enter data and print numerical and/or graphical report.

b) Analysis - compute calculations, develop reports.

c) Synthesis - Analyze data, examine cause and effect, make predictions, discuss "what-if."

d) Evaluation – Perform a simulation, modify parameters, experiment with different data, discuss impact of the parameters involved, reach conclusions, develop or refine a model, develop recommendations or guidelines, reach decisions.

Most of the spreadsheet usage was classified into the application or analysis categories. There was little overall evidence of higher order learning activity occurring. The spreadsheets were used as emancipatory tools, saving time in such tasks as graphing. Few instructors had been able to integrate spreadsheets into the curriculum at a higher level of learning. They noted that students were concerned with getting through the assignment quickly by learning how to manipulate the software to produce the desired results. However, the researchers noted that few teachers were using the spreadsheets at a higher cognitive level, thereby limiting the level of learning.

Lithner (2000) found that most textbook exercises can be solved based on what is familiar and remembered, rather than on even elementary mathematical reasoning and accuracy, without understanding the core mathematics content contained in the chapter. It was found that when students are working with textbook exercises, their mathematical reasoning consisted of strategy choices and implementations that are carried out without understanding the intrinsic proprieties of the components involved in the solution. In order for this strategy to work, the students needed only to find solution procedures to copy; their understanding of the copied steps was thus localized, and they mad little or no attempt to construct their own solution.

Learning Methods and Materials

People learn to develop spreadsheets through formal training, individualized learning using on-line tutorials, manuals, or peer instruction. Some of the common methods used to learn how to develop spreadsheet applications are learning-by-example,

using tutorials, problem-solving, exploratory learning, means-end analysis, and trial-and-error (Reimann & Neubert, 2000; Charney et al., 1990; Gick, 1986). The first three are the most common used in formal spreadsheet training environments and are based on analogical reasoning.

Charney et al. (1990) considers learning to use computer applications to be skill learning consisting of three components: knowing what procedures exist for accomplishing various goals, selecting the right procedure for a given circumstance, and knowing how to execute the procedure. Charney et al. (1990) contended that these three components are generally not supported in learning materials. The methods of learning computer applications analyzed are as follows: learning-by-example, using tutorials, and problem-solving.

Learning-by-examples.

One method of reasoning commonly used in computer skills instruction is learning from examples. Learners use analogical reasoning when solving the first problem in a domain and/or with a new tool. Referring to solutions provided as examples is preferable to search based problem solving, where the learner uses isolated, poorly understood operators. An important cognitive process in learning-by-example is self-explanation while reading texts or worked-out examples. Self-explaining includes activities of monitoring one's understanding as well as of engaging in knowledge construction in order to overcome self-diagnosed problems of understanding (Reimann & Neubert, 2000).

Gick (1986) noted that using worked examples provides an opportunity for people to study solutions rather than just to follow solution procedures. The process of comparing worked examples may contribute to the abstraction of the problem schema. A learner compares problems and abstracts the common features of both. Mapping between elements of two domains may help in the development of entities that are abstractions over the domains. The more general and context free the examples are, the more likely analogical transfer is to occur to a new problem. Clarke, Ayres, and Sweller (2005) found that worked examples were effective for novices, but suggested that as expertise develops a problem-solving approach may be more beneficial.

Many instructional texts are based on learning-by-example, consisting of numerous worked-out examples to study and use as examples to solve problems. The worked-out example gives an explicit set of steps to follow to perform a certain procedure towards a specified goal. Rules for use might be included in the text as well as descriptions of ineffective uses or common misuse. Examples included in computer learning materials help learners categorize problems with similar solutions and construct solutions to novel problems by analogy to the example. Learning-by-example is a common approach for computer user manuals, illustrating features or commands.

Problems with learning-by-example include the possibility of low motivation due to the passive nature of following examples. In addition, Charney et al. (1990) note that the second and third components of skill learning are not included in the learning process. Learning-by-example might help learners form declarative representation of concepts,

procedures, and problem situations in a domain, but it may not provide enough independent practice at selecting and applying the procedure.

Reimann and Schult (1996) reported on problems that novices have when using learning from examples. Problems must be interpreted and translated, a process often demonstrated using examples. It was also difficult for novices to decide which step to take next in the translation process. Analogical transfer may not take place because students fail to recognize structural similarities. Even when an example is explicitly provided, mapping to the correct representation and modifying the solution are difficult. Students often map syntactically from the example to the result, leading to ineffective and possibly erroneous solution attempts.

Using tutorials.

Interactive tutorials, like examples, give the learner a problem and a series of steps used to solve the problem (Gick, 1986). The learner enters the steps of the solution on the computer and is given feedback and messages. This would appear to have the same benefits as learning-by-example but without the passiveness. However, learners can enter keystrokes mechanically, without connecting them to the overall process. The second and third components of skill learning are still missing when using interactive tutorials.

Problem solving.

Gick (1986) describes problem solving as a learning strategy that starts with a text containing prepared problems and is adapted from chapter-end exercises in math and science textbooks. Instead of studying or carrying out a solution prescribed by the book,

the learner develops a solution by retrieving possible procedures from memory, setting appropriate procedures, generating correct instantiations, and executing them correctly. Feedback is often provided as solutions in the back of the text. This method has the advantages of worked out examples in that the problems provide information about the skill domain. The learner can examine and compare problems and make inferences regarding problem situations and solutions and mentally store solutions to be used as models.

If the problem only appears at the end of the chapter that contains the content about the procedure, learners are probably only applying the procedure, and not selecting between procedures. Problem-solving, if introduced this way, does not contain the second component of skill learning. Learners may spend time on incorrect solutions and not develop good models of solutions.

Case-based learning.

It is often easier to learn by retaining a concrete experience than by trying to generalize from it. Case-based reasoning proves to be an effective method for learning (Kolodner & Guzdial, 1999). In case-based learning, the solution to a previous case is reused. After the new solution is developed, it can thus be evaluated for correctness. If the new solution is successful, learning takes place. If the solution is not correct (an error), learners can repair the case solution using domain-specific knowledge. If errors can be included in case information, they can be used in future retrievals. In fact, errors may be considered necessary in case-based learning. Students who are using case-based reasoning need to attempt to apply what they think is applicable and fail in that in order

to know to focus attention on subtleties that they had not previously been aware of (Kolonder & Gudzial, 1999).

Means-end analysis.

When learners are not using analogical transfer, they are often using means-end analysis as a way of solving the problem. This strategy is often the choice for novices. Means-end analysis does not help in learning problem states and associated actions that result in new states (Gick, 1986). In addition, the focus on states and associated actions imposes a working memory load that may hinder the development of schema. Preventing the subject from using means-end analysis may result in better learning of the structure of the problem. A goal–modification, such as finding the values of all variables instead of just what appears to be the next state, may improve problem-solving, increase the use of forward thinking strategies, and help in correct categorization of problems (Gick, 1986).

Trial and Error.

Reimann and Neubert (2000) reviewed the research on computer skills learning. New users of highly interactive software rarely have plans. They lack information about primitive actions and their application. They also do not know how to decompose a goal into subgoals. Novices need to create goals dynamically, but they also need to know which actions are available. They may have problems mapping from the current goal to a goal that they can get to from where they are. This *promotes* use a type of trial and error learning described as modified "hill climbing". Hill climbing involves the calculation of the next state for each applicable problem solving operator and the calculation of the difference of those next states to the goal state. The operator resulting in the minimal

difference will be selected and applied. Hill climbing is similar to means-end analysis. Since novice users do not know about the effects of actions on the environment, the selection of an action is based on a different heuristic to get beyond trial and error. Novice users often use the "label observed heuristic," a tendency to select menu choices whose prompts are terms with the description of the task.

Over time, users will make fewer errors, do less backtracking, and spend less time in the environment because they learn from problem strategy. Weak hill climbing is eventually replaced by knowledge-based strategies. The environment will still mainly trigger subgoals, but terminal actions will be known and will not have to be discovered. Effective acquisition of action knowledge requires having a clear representation of the effects that an interface action causes, which is not trivial in complex graphical environments. In such environments, many changes can occur simultaneously, some relevant for the main event of the action and some not. A single action may also have no visible effect, and two or more actions are needed to experience a noticeable change in the environment. Examples in spreadsheet learning would include formatting changes, data changes in one cell with cascading effects, hard-coding, changing a formula. but not copying. In addition, due to the physical layout of a spreadsheet, some change may take place beyond the viewing areas. Thus, trial and error-based learning is not an effective way to learn about complex software. Studying the manual first might be a better approach, but is hardly ever used (Reimann & Neubert, 2000).

Summary

Charney et al. (1990) noted that few studies had directly compared the strategies such as learning-by-example and problem solving for computer skill learning. Means-end problem solving strategies used by novices impose a heavier cognitive workload. When examples required students to integrate information from several sources, such as textbook and computer, the additional cognitive load canceled out the benefits. When learning computer applications, problem solving required more training time than learning-by-example and tutorials, but commands learned through problem solving were more likely to be selected and applied correctly. In addition, the types of problems might matter in learning-by-example and studying worked out examples in algebra was beneficial.

The first strategies described above (learning by example, using tutorials, problem solving, case-based) are indirect methods of accelerating schema learning (Gick, 1986). They suffer from the learner having to induce strategic knowledge from previous examples. One problem is that the strategic knowledge may be associated with the domain and may be difficult to explain. The second is that the direct strategy may not work for all, yet learners may solve some problems using the means-end approach describe below.

One concern with any of these analogical approaches is that a teacher-provided template is usually necessary for students to generate an adequate spreadsheet representation of a problem. Research on analogical reasoning has shown that subjects often fail to use previous problem solving experiences without hints. Often, the base

problem has to be indicated explicitly (Schieter & Gerjets, 2002). Another significant issue is that a result can be obtained in the absence of learning (Gick, 1986).

Cognitive Theory

The reasoning and learning involved in developing spreadsheet applications and in the errors created can be described via cognitive theory, as well as via error making and motivation. Cognitive theory presumes that knowledge and learning involve control modes and cognitive structures. Cognitive activities are guided by complex interplay between attentional (controlled or conscious) processing and schematic (automatic or unconscious) processing. Cognitive structures are working memory and the knowledge base, long-term memory. Working memory is associated with the attentional control mode, and the knowledge base is associated with the schematic mode. Knowledge is stored in schemata, higher-order, and generic cognitive structures. The encoding and representational functions include lending structure to perceptual experience and determining what information will be retrieved from memory (Reason, 1990).

Learning

Mayer (2002) defines learning as a relatively permanent change in someone's knowledge based on the person's experience. Meaningful learning is learning that leads to transfer, that has an effect on new learning. If the learning amounts to acquiring a collection of specific responses to situations, meaningful learning does not take place.

Cognitive and constructivist perspectives emphasize what learners know and how learners think about it as they actively engage in meaningful learning. Learners bring with them previous knowledge, personal goals, and prior experiences. In instructional settings, they use all of these to make sense of the information they encounter to construct their own meaning. This constructivist process involves activation of prior knowledge and cognitive processes that operate on that knowledge (Anderson & Krathwohl, 2001). In constructivist learning, students engage in active cognitive processing, such as paying attention to relevant incoming information, mentally organizing the information into coherent representation, and mentally integrating incoming information with existing knowledge (Mayer, 2002).

Mayer (2002) describes meaningful learning in terms of cognitive theory. The learner is actively engaged in the process, forms hypotheses, tests the hypotheses, keeps correct hypotheses, and selects new ones for incorrect answers. Learning involves making sense of the learning situation, and feedback helps build rules or procedures. It is not the feedback itself that promotes learning, but the learners' interpretation and understanding of the feedback. Students engage in meaningful learning when they build intrinsic motivation that supports a self-monitored change in behavior, rules that support concept learning, and general procedures that support skilled performance. Meaningful learning is consistent with the view of learning as knowledge construction (Anderson & Krathwohl, 2001).

Errors

Cognitive theory can be used to explain error making. Schemata are high-level knowledge structures that contain information variabes. Each variable accepts a particular kind of information. If current inputs fail to supply specific data for these variables, they take on default assignments, values derived from past transactions (Reason,1990). A schema contains evidence of how a particular recollection should appear, not a representation of what it should *not* look like. Systematic errors can result from fitting the data into the wrong schema, employing the correct schema but filling gaps with best guesses, and on relying too heavily on active or salient schemata.

To help explain errors, Reason (1990) describes the skill-knowledge-rule framework for cognitive control. The three levels of performance: skill-based, rule-based, and knowledge-based-correspond to increasing levels of familiarity with the environment or task. At the skill-based level, human performance is governed by stored patterns of preprogrammed instructions represented as analog structures. In the rule-based level, familiar problems are solved using stored rules or productions of the type : *if (state) then (diagnosis)* or *if (state) then (remedial action)* . Errors at this level are associated with misclassification of situations leading to the application of the wrong rule or an incorrect recall of procedures.

At the knowledge-based level of performance, errors arise from resource limitation and incomplete or incorrect knowledge. With increasing expertise, the primary focus of control moves from the knowledge-based towards the skill-based levels. All three performance levels can co-exist at any one time.

Motivation

Motivation can also be addressed through cognitive theory and meaningful learning. Students can be motivated based on interest, self-efficacy, or attribution. Students' motivation is based on how they interpret the learning situation. Wanting to learn is related to one's beliefs about learning. The motivation to learn may depend on how the student thinks about the personal relevance of the material, his own confidence, and if he believes that hard work will result in success. The learner is a decision-maker who bases actions on interpretation of incoming information (Mayer, 2002).

Students' perception of mathematics, spreadsheets, and the course may affect the accuracy of spreadsheet applications they develop. The perception of the spreadsheet as a useful tool for data analysis, modeling, or problem-solving will impact motivation to learn. Since part of the reasoning students are using is converting an algorithm into a suitable mathematical representation, they may associate the learning with their experience in mathematics courses. Students' perception of the content, value, and instructional approach of a formal course may affect how well they learn the concepts. Lambert and McCombs (1998) note that student motivation can be attributed to the knowledge of content, perception of instructional priorities, and perceived climate in the classroom. While motivation may be a factor in spreadsheet accuracy, a full investigation of this topic is beyond the scope of this study.

Research Methods

Qualitative and quantitative methods have been used to investigate spreadsheet phenomena and reasoning. Most spreadsheet error studies were quantitative studies that analyzed data statistically (Tukiainen, 2000; Panko & Sprague,1999; Panko, 2005; Janvrin & Morrison, 2000). One of the goals in most cases was to count spreadsheet errors in some way. Kruck et al. (2003) used quantitative methods to determine change in cognitive processes due to learning to develop spreadsheet applications. Results in some studies are often reported as success rates or performance based on accuracy, usually some measure of correctness. For the purpose of this study, the presence of errors may be indicated in research by incidences that were not successful, reductions in performance, or inaccurate spreadsheets.

Qualitative methods were used when the goal was to understand what the participants were thinking. Interviews were used by Nardi and Miller (1991) and Hendry and Green (1994) to explain reasoning used in spreadsheet development. Lithner (2000) used interviews to try to determine what participants were thinking after they had developed a spreadsheet or solved a word problem. Wilson et al. (2003) and Wallace et al. (2002) used think-aloud protocols to investigate what people were thinking while developing spreadsheets. Lithner (2000) used similar methods to investigate reasoning while solving calculus word problems. In general, the think-aloud protocol provides an opportunity to find out what people are thinking while performing a task. Observations of participants and researchers notes were also used to infer reasons for behaviors in several

studies (Nardi & Miller, 1991; Hendry & Green, 1994; Brown & Gould, 1987, Wilson et al. , 2003; and Wallace et al.,2002).

Anderson (1989) and Wallace et al. (2002) retained electronic copies of interactions with the computer. Anderson used the information to trace the steps used in solving a problem. The electronic information was used to measure errors

As demonstrated by the research reviewed, different types of errors occur in spreadsheets (Panko, 2005; Rajalingham et al., 2001), error rates can be measured (Panko, 2005), and errors can be found (Panko & Sprague, 1999; Panko & Halverson, 2001). Certain types of reasoning are known to be better than others are and may directly imply fewer errors. An important category of errors is caused by faulty reasoning. It is possible to study reasoning used during spreadsheet development. It is also possible to observe and study errors being made. Subjects may be able to state or explain the reasoning they used. Based on the studies by Wallace (2003) and Lithner (2000), participants may also be able to give insight into how they learned the reasoning that allows errors to happen. These might also prove to be elusive problems, but the reasons for that may be useful.

Summary

The research on spreadsheet errors indicates that spreadsheet applications are often inaccurate and that errors are diverse in nature, difficult to find, and difficult to prevent. Most research efforts to reduce spreadsheet errors are based around tools designed to find errors, on testing techniques, and on adaptations and extensions of the current spreadsheet model based on programming languages. Little has been done to

reduce the initial errors being made and particularly those resulting from lack of understanding of either the problem or spreadsheet capabilities.

Error classification systems have been developed to help determine what kinds of errors are being made and how they can be dealt with (Panko, 1998; Rajalingham et al., 2001). Spreadsheet errors can be categorized into either of two major categories, quantitative and qualitative. Quantitative errors produce incorrect numeric results. Errors that change the interpretation of the spreadsheet application or impact future maintenance are considered qualitative, including hard-coding and labeling errors.

Several taxonomies have been developed to categorize spreadsheet errors. The most extensive and detailed was proposed by Rajalingham et al. (2001) and includes categories for reasoning errors. These errors are a result of to flaws in thinking. Research on reasoning used during spreadsheet application development indicates that this is a complex cognitive task, requiring juggling of domain, logical, physical, and spreadsheet models in conjunction with goals and plans. The spreadsheet is more closely aligned with the problem domain than traditional programming languages and, as such, spreadsheet applications are conceptually easier to develop than computer programs. However, some aspects of spreadsheet development are difficult and can lead to errors. There are some basic skills used in spreadsheet development which can be difficult to master and thus errors result. Referencing techniques are difficult, particularly for novices (Tukiainen (2000). Whittle and Janvrin (2002) found that naming cells reduces errors due to cell referencing. A copy or fill operation in a spreadsheet can modify a cell reference, introducing errors, according to Brown and Gould (1987). The copy operation may

require absolute referencing, which is a difficult concept for students. In addition, copying an incorrect formula increases the inaccuracy of the spreadsheet. If the developer does not use the copy capability, they may enter formulas individually for each cell in a series. This approach results in increased typing, and accidental or mechanical errors result from mistyping (Panko, 2005). It would follow that any process that resulted in an increased amount of typing can lead to accidental and mechanical errors.

The research on the relationship between user attributes and spreadsheet errors has thus far indicate there is little or no association between factors such as age, gender, major on error rates. Also, spreadsheets developed by both experienced and novice users have similar error rates and a diversity of errors that would make automated error-discovery difficult.

Errors can be a result of flaws in the reasoning process. Several different types of reasoning strategies are used in spreadsheet development. Reasoning such as means-end, trial-and-error, and imitation may lead to errors. Analogical reasoning and case-based reasoning can be effective, but if there is a flaw in the transfer from previous problems to the current situation, an error can result. Omission-type errors can result from cognitive overload due to the developer trying to keep track of cells, formulas, mental models, and the current state of the spreadsheet. Although research in reasoning that leads to spreadsheet errors is scarce, there have been efforts in similar areas such as mathematical word problems and algebra.

Errors created during spreadsheet development may be a result of how the developer learned to reason about spreadsheets or the domain. People learn to develop spreadsheets in a number of ways including formal courses, peer training, tutorials, self-learning, and manuals. Domain knowledge could come from previous courses or work and life experience. The methods and reasoning people use to learn how to solve problems using spreadsheets may be significant factors in the accuracy of the resulting spreadsheet application.

Research efforts to prevent errors include instructional techniques, peer assessment and audit, design techniques, and including validation criteria and boundary checking in design and development. Chadwick and Sue (2001) suggested that instruction in the impact of errors, self-audit, and peer assessment may help students understand errors. Janvrin and Morrison (2000) and Whittle and Janvrin (2002) suggested design diagrams and cell naming to reduce referencing errors. Software add-ons can provide mechanism for data validation and discovery of errors in units and referencing. Wallace et al (2003) developed software to include conditions in cells along with data and formulas. These conditions (assertions) give developers the capability to communicate aspects of their mental model of the problem beyond what is in the formulas and to provide a method to reason about the integrity of the logic. Abraham and Erwig (2006) developed a system which a developer can generate a template for spreadsheets and a set of correct update operation, which may help prevent certain kinds of errors such as range, reference, and type errors.

Wallace et al. (2002) reported several intesting behaviors by novices during spreadsheet development. Subjects tended to use a "local testing" strategy, testing only the information in a single cell, for example, checking math. Tukianien (2000) also noted that novices think locally. Wallace el al. (2002) noted that subjects also avoided testing using "easy" inputs, which would have allowed them to use their domain knowledge to predict and easily determine the accuracy of their answers. The students viewed these inputs as unrealistic test values or felt that the computer could do them by default.

As someone is learning to develop spreadsheets to solve problems, recognizing errors may not have an effect on learning if the individual is not concerned about the impact of errors. Familiarizing novices with others' mistakes and misconceptions might reduce the number of errors. Chadwick and Sue (2001) proposed that introducing risks associated with incorrect models might also encourage students to apply error-checking controls during development. Students are generally taught how to do things, and perhaps part of instruction should include how *not* to make errors. Brown and Gould (1987) infer that the consequences of errors in an academic exercise are not as significant as those costing substantial sums of money in a business or industrial situation.

Finding errors in existing spreadsheets is difficult and not very effective. People tend to find about half the errors (Galletta et al., 1997 ; Panko, 2005). Spreadsheet users and developers are often overconfident with calculations generated by the spreadsheets (Panko, 2005) and may have trouble spotting errors in their own spreadsheets (Chadiwck & Sue, 2001).

Spreadsheet reasoning, learning, errors, and motivation can be discussed in terms of cognitive theory. When people learn, they use conscious processing with working memory and unconscious processing with long term memory. Working memory holds information about new concepts and current problems. When learning about spreadsheet concepts, the student is actively engaged by interpreting new information and feedback, forming and testing hypotheses, and constructing new knowledge. Depending on familiarity with spreadsheet concepts, people will perform at one of three levels: skill, rule-based, and knowledge. Errors can occur at each performance level and the performance levels can co-exist during mental processing of a situation. Feedback helps build rules or procedures and the learners' interpretation and understanding of the feedback promotes learning (Mayer, 2002).

Knowledge is stored in long-term memory as schemas. When someone learns concepts involving spreadsheet development schemas are created. If a spreadsheet developer associates a problem with an incorrect schema, errors can occur. Later, the developer can retrieve the information and apply it to a situation. However, Reason (1990) notes that a schema contains evidence of how a particular recollection should appear, not a representation of what it should *not* look like. If the developer cannot match a schema to a particular problem or mismatches a schema to a problem, errors can occur.

As the learner is an active participant in the spreadsheet learning process, they choose how to engage in the learning process based on their perceptions of the domain, value, and instructional approach used. In a spreadsheet course, the content, pedagogy,

and classroom environment of the course as well as the value they place on knowing how to develop spreadsheets may affect how students learn to reason when developing spreadsheets. These reasoning processes used in the development of spreadsheet applications will have an effect on the accuracy of the resulting spreadsheet.

The spreadsheet becomes a cognitive tool for both problem-solving and learning. The spreadsheet relieves the developer of the burden of mathematical calculations and provides a flexible environment for reasoning. The developer learns to select and approach from a variety of potential solution methods and apply it to the problem. In a learning situation the student can integrate success, failure, feedback into their knowledge base and refine iterative approaches to problem solving.

Spreadsheet learning and errors have been investigated using both qualitative and quantitative methods. Field studies as well as lab studies have been used to gain information. The factors commonly analyzed are error rates, error types, and error detection in existing spreadsheets. Little has been done to investigate the initial causes of errors although spreadsheet developer thought processing has been examined using computerized methods to capture steps in problem solving, interviews and think-aloud protocols.

There are limitations to many of the research efforts on spreadsheet errors. In the lab-based studies, a student sample with a small spreadsheet is not representative of industry spreadsheet users and of developers' experiences. Many factors studied were self-reported and may not accurately reflect the persons' backgrounds. Length of time people have used a spreadsheet does not reflect on the quality of their experience.

Chapter III

Methodology

This study was designed to investigate students' reasoning when learning to create spreadsheet applications and the reasoning that results in errors. In particular, the overarching problem for this study was to investigate students' reasoning when creating spreadsheet applications, to see how that reasoning develops, and the impact of the reasoning used that results in errors in the spreadsheet solutions. The design of the research was focused toward answering the following research questions:

1. What is student's reasoning that results in *domain knowledge* errors in spreadsheet solutions?
2. What is student's reasoning that results in *implementation* errors in spreadsheet solutions?
3. What learning experiences shape students' reasoning strategies that result in either correct or incorrect solution to problems?

The theoretical framework for investigating students' reasoning associated with spreadsheet errors relied on the spreadsheet reasoning categories described in the taxonomy developed by Rajalingham et al. (2001) and on cognitive aspects of student learning. The reasoning, learning, error-making, and motivation in terms of cognitive theory and constructivism were described in Chapter II.

Spreadsheet reasoning errors are manifested by entering incorrect formulas. Subcategories of reasoning errors are *domain knowledge* and *implementation. Domain*

knowledge errors are due to a lack of understanding of the problem needed to accurately model the problem solution. Included in this classification are errors resulting from selection of an incorrect algorithm (*real-world knowledge*) and inaccurate construction of a formula to implement a correctly chosen algorithm (*mathematical representation*). *Implementation* errors are due to lack of knowledge of the full use of the functions and capabilities of a spreadsheet package in use, not understanding of the spreadsheet principles, concepts, constructs, reserved words, and syntax. *Implementation* errors are further divided into *syntax* errors and *logic* errors. *Syntax* errors occur when the formula contains characters and symbols that are not recognized by the spreadsheet for performing the desired function. Most spreadsheets have the capabilities for catching these errors. A *logic* error produces incorrect values resulting from an incorrect construction of a formula; this error may be due to a lack of understanding of the specific features and functions of the spreadsheet and thus producing an incorrect value. Implementation errors include copying and referencing techniques that result in incorrect values. As mentioned in Chapter I, for the purpose of this study, qualitative maintenance errors as described by Rajalingham et al (2001), such as copying and referencing techniques that may lead to a future error, but do not currently result in an incorrect value, are also considered.

As noted in the section on cognitive theory in Chapter II, these reasoning errors can be explained using cognitive theories of learning. Students achieve meaningful learning through an understanding of underlying principles and concepts. They construct their own understanding based on prior experiences with spreadsheets, the mathematics

involved in spreadsheets, the information presented in their learning experiences in a spreadsheet course, their learning style, and the classroom learning environment.

For the purpose of this study, a student's domain knowledge and reasoning was the combination of knowledge prior to the course and the skills, knowledge, and reasoning gained from the course. Since most spreadsheets that are developed are mathematically-based, a student's background and ability in mathematics could be associated with domain knowledge and therefore with real-world and mathematical errors, as described in the taxonomy (see Figure 1 in Chapter I). For example, students may have gained spreadsheet skills in a previous course, from peers, or through self-instruction. They may also be overconfident in their own skills, abilities, and experience with spreadsheets. These experiences then impacted their reasoning with spreadsheets during formal instruction on spreadsheet skills and concepts.

This chapter describes the methodology used in the conduct of this study. In the first section, the design of this qualitative phenomenological study will be described and the think-aloud protocol used will be introduced. The next section, Participants and Study Setting, describes how college students enrolled in a five-week introductory spreadsheet course took part in this study. Establishing the design and participants gives a foundation with which to describe the spreadsheet tasks that were used in the think-aloud sessions in the next section, Spreadsheet Tasks. The preceding sections provide the background needed to identify the data sources and their purpose in the study.

The next two sections describe the data collection and data analysis. The schedule for data collections and format for think aloud sessions is detailed in the section

on data collection. The last section describes data was compiled into useful information, including transcription of data sources, identification of errors and reasoning, and triangulation. Finally the association between the data analysis and research questions is described.

Design of the Study

A qualitative phenomenological case study approach is used in this investigation. A qualitative study enables the researcher to gather descriptive information about the reasoning processes that are expected to be complex and to vary among individuals. A phenomenological approach allows the focus to be on an exploration of how students make sense of their experiences, transforming those experiences into their consciousness (Patton, 2001). Patton suggests the following foundational questions for research on how individuals construct their understandings.

- How have students in this setting constructed reality?
- What are their reported perceptions, explanations, and beliefs?
- What are the consequences of their constructions for their behaviors?

This research was designed to identify responses to these questions as they pertain to the process of creating spreadsheet errors. Case studies were developed by examining individuals' thought processes as they develop spreadsheet applications with a focus on circumstances that typically lead to spreadsheet errors. A think-aloud protocol was used for investigating each student's thought processes (Patton, 2001). In a phenomenological approach, in-depth interviewing of the students who are directly experiencing the

phenomenon is the methodology that best captures and describes their experiences. A retrospective report in the form of an interview was also conducted after the student's spreadsheet development in order to probe reasoning associated with the error-prone sections.

Each of the three case studies included the information from the classroom observations as the students are learning about spreadsheet skills, concepts, development, and design as well as the transcripts of think-aloud sessions as the participants solved worked on spreadsheet tasks specifically designed for the study, and interviews with the participants after the think-aloud activities. In addition, material from classroom exercises, homework activities, and exams were included in the case studies. A background survey gave information regarding the student's prior experience with spreadsheets and mathematics. Once assembled, the case studies were compared to identify reasoning processes, common errors, reasoning processes that led to errors and those that support dependable results, as well as learning experiences that had been identified as shaping students' reasoning strategies.

Participants and Study Setting

Volunteers were solicited from students who were enrolled in an introduction to spreadsheets course at small four year college during spring term, 2007. This lab-based one credit hour course met once a week for a single two-and-a-half hour session for five weeks. Twelve college students with varying backgrounds were enrolled. The course curriculum and instruction engaged students in learning through examples from the text

with step-by-step instructions for the specific tasks. Applications were developed for hypothetical situations based on the textbook chapter content. The researcher, in the role of a researcher/instructor, demonstrated new spreadsheet techniques and pointed out concepts that often cause problems during the classroom instruction. During of each class session, the instructor interacted individually with students, providing feedback on work completed while also discussing difficulties and applications of the concepts involved with individuals and small groups. The instructor also designed and assigned summary activities to integrate learning and assessment tools.

The minimal criteria for participation in the study was that a student was enrolled in the course and agreed to participate. If the number of volunteers was large ($n>6$), the selection process included criteria for selecting students with varying degrees of spreadsheet accuracy to try to better discern reasoning methods that were effective versus those that were not. The researcher/instructor asked for volunteers from the class for the first two weeks of the course. Participants in the study were offered extra credit in the course. An alternative activity for extra credit was also offered to those not participating in the study. Three students volunteered for the study and met the minimum criteria for participation. Two of the students were traditional female students while the third was a slightly older male with a military background. All had little or no previous spreadsheet experience and a standard mathematics background (college algebra).

The instructor noted after the first two weeks of class that the learning environment was not effective. Students spent too much of the class working on the step-by-step activities in each chapter. There was little or no interaction in the class, even

during the chapter review activities that were far less structured. Students worked at different paces and thus were often focused on different sections of the content at any given time. Review activities were assigned as homework. In week three, the format was changed so that the step-by-step sections of the chapters were assigned as homework and class time was used for more comprehensive and complex activities and for group activities. The interaction increased between students and between students and the instructor. Students seemed more engaged in the learning process with more opportunities to integrate the material into their understandings.

Think-aloud sessions were arranged with each participant outside of class time over the duration of the course. The goal was to have three sessions with each participant at different phases of the course to observe whether his or her reasoning was consistent or changed during the course. Sessions took place in the last three weeks of the five-week course. The setting for each session was similar to the environment in the lab-based course.

Spreadsheet Tasks

The spreadsheet development tasks (Appendix A) used for the think-aloud study were developed by the researcher based on the course content and expectations along with a consideration of common errors made by students in spreadsheet applications. The criteria for the tasks were as follows:

- The spreadsheet skills needed for the solution to the task were based on content and skills in the course up to the point of the think-aloud session in

which they will be used. The interest in this study was to determine whether and how students in the course developed particular reasoning strategies that lead to errors during the learning processes associated with the spreadsheet course. The tasks needed to:

- Reflect the course material covered. This introductory course covered common topics such as formatting, graphing, formulas, functions, decision making, and copy and fill operations.
- Include copy and referencing spreadsheet techniques. These concepts tend to be difficult for students and are often the source of errors.
- Provide the capability to distinguish between imitation and understanding.
- Be relatively simple in terms of size and time to complete as to not overwhelm the subjects and add stress as a factor. The task was designed to take most students the time allotted in Table 3 yet no fixed time limits were set on the actual tasks.
- Be clearly described and easy to understand to avoid adding to the cognitive overload (Kintch,1998).
- Involve the reasoning described by Rajalingham et al. (2002). Therefore, the tasks should include formulas and functions that involve real-world knowledge, mathematical knowledge, and spreadsheet skills, creating a situation where students might make either domain knowledge errors or implementation errors.

- The domain knowledge needed to be consistent with the domain knowledge in the spreadsheet course content and familiar to the college students (for example, the number of days in the months, days in a year, measurement units, the melting points and freezing points, or volume and profit formulas).

In order to establish content validity, the tasks were reviewed by an education spreadsheet and mathematics expert, a spreadsheet instructor, and an instructional designer. These individuals are aware of the spreadsheet knowledge expected from the course and the types of activities that are commonly used for learning and assessment. The tasks and instructions were revised based on recommendations from these individuals. The instructional designer reviewed the tasks for content, readability, and format; revisions were made based on the results.

Data Sources

Multiple data sources were used to investigate the students' reasoning during spreadsheet creation and error making. These data sources included think-aloud sessions, student interviews, researcher observations during the classroom activities as well as during the think-aloud sessions, researcher/instructor journal, and student background surveys. Researcher/instructor field notes from classroom observations, think-aloud sessions, and follow-up interviews provided mechanisms by which to learn about the

reasoning used when students were making domain errors and implementation errors, as well as to speculate about how that reasoning was learned or acquired.

In order to identify what reasoning students use during the process of translating a problem into a spreadsheet application, it was necessary to observe the students' working process and interact with them during the process. Domain knowledge errors occur during initial formulation stages of the process as the students were mentally developing an application. Implementation errors took place as an application was being developed. In the classroom setting, it was possible to observe and ask the students about certain approaches they were using in creating their spreadsheet solutions to problems. But, to identify the reasoning students used in forming various mental models of problems and spreadsheets, it was necessary to gather more information than a classrooms setting allowed.

Ericcson and Simon (1993) described two forms of verbal reports that are the closest reflection of cognitive processes, concurrent verbal reports and retrospective reports. Think-aloud reports were described as concurrent verbal reports in which cognitive processes were verbalized directly. Retrospective reports provided a method of accessing information from short-term memory immediately after a task completion. Think-aloud sessions were conducted as the participants worked on spreadsheet tasks to capture their reasoning during spreadsheet development. To probe particular areas of interest in reasoning and errors,a think-aloud session was followed by a retrospective interview. Additionally, researcher/instructor classroom field notes and post-think aloud interviews with students provided information about how the students learned reasoning

that may have led to domain and implementation errors during spreadsheet application development.

Think-aloud sessions

Think-aloud studies have been used previously in computer spreadsheet investigations (Wallace et al., 2003; Wilson et al., 2003). Such studies were designed to monitor thought processes recording the reasoning used (i.e. "Now I need to add the rows because …", "Use the sum function because …", "This is like the problem we did before, only the term is in months, instead of years"). The goal of using this approach was to elicit the inner thoughts or cognitive processes taking place as the students worded on the spreadsheet tasks (Patton, 2001). While the students worked to develop spreadsheet solutions to the particular tasks, they were asked to verbalize the thinking that led them to take particular actions with the tasks. A practice problem was used to introduce the think-aloud protocol during the first session (Appendix A).

Three think-aloud spreadsheet tasks were developed as described previously (see Appendix A). During each think-aloud activity, if the student faltered in thinking aloud or stopped, the researcher prompted with questions such as "Keep talking", "What are you thinking", "Why did you do that." Also, in the event that a student was not sure how to proceed or to approach the next subtask, the researcher encouraged them to develop their best solution and continue working with the problem. If a student became stuck, the researcher/instructor assumed an instructor role and tried to encourage progress through questions that might lead them to more clearly identify their problem and/or relate it to material learned in class. These instances also gave information on how the student

reasoned and what problems they had in developing a solution. At the same time, these opportunities provided teaching and learning opportunities for both the students and instructor. When students were unsure of a solution, a discussion was initiated about skills, reasoning, learning, and background. The sessions were videotaped and later transcribed for detailed analysis.

Student interviews

Student interviews were used to develop retrospective reports of the students' reasoning during development of the application. The purpose of the interviews was exploratory, verifying a hypothesis, or determining the limits of its generality (Krathwohl, 1998) and it was important to use approaches that obtained a response close to natural conversation. In this study, the purpose was to explore students' reasoning through a partially structured interview conducted by the researcher.

A standard protocol of questions was formulated (Appendix B) but the researcher directed questions during the interview as deemed necessary to focus on each individual student's reasoning and error-making. The questions were open-ended and the researcher encouraged participation by expressing the usefulness of the responses. More specifically, the researcher questioned the students about the thinking involved in the selection of the formulas, functions, and referencing. Further verbal probing was used to determine how that reasoning developed.

The students were questioned about the accuracy of the spreadsheet they created since often spreadsheet developers are confident about the accuracy of their products (Panko, 2005). In general, although they were not sure the spreadsheet developed was

completely accurate, the students were not aware of how many errors existed in their solutions. The researcher pointed out errors and asked whether they could describe what the problem might be. If they could not determine the problem, the researcher indicated the problem and asked about a solution. Then the student and researcher discussed the correct formulas and focused the questions on cells containing errors to encourage the students to explain their reasoning during the design and development of those cells.

Further questions sought to identify how the student acquired the ideas that led to the errors. In some situations, the participants indicated they did not understand the problem or how to go about solving it. The researcher then tried to ask the student questions to determine what part of the problem or solution process they had trouble with and why. At other times, the participants sometimes indicated they were trying to match it to a problem done in class and the researcher would ask about that problem or how that problem applied to the current one. At times, the participants stated that they were not good at mathematics or that mathematics was the problem. The researcher would then try to reframe the current problem into one they might be able to solve.

If the student has made the same error on previous activities, the researcher tried to find out not only what the faulty reasoning used is, but also why they continued with a reasoning strategy that leads them to inaccurate results. In cells where students produced an accurate spreadsheet with no errors, the researcher asked about reasoning regarding areas of the spreadsheet that are prone to errors. The objective was to determine what reasoning led to correct results as well as to errors.

All interviews were video-recorded and transcribed verbatim and all work of each participant was assembled to support the transcriptions. Electronic versions of the spreadsheets created during the sessions were maintained for analysis. Errors were categorized using the reasoning portion of taxonomy described in Chapter II. This categorized information was used in the analysis of students' error patterns.

Researcher/Instructor field notes

The researcher/instructor maintained a set of field notes during each of the classes and the think-aloud sessions. The purpose of the notes was to gather data that focused on the students' reasoning and learning. The researcher as instructor interacted with students during the class period, performing formative assessments and providing feedback about their work and ideas. The researcher/instructor's observations and interactions included summary observations of the entire class as the students learned to develop spreadsheet applications. After participants volunteered for the study, the researcher/instructor focused his observations on the participants' actions in the classroom environment as they worked from both observations and conversations. The protocol used as a guideline for classroom observations is included in Appendix D.

The notes during the think-aloud session were used to help identify reasoning processes and error-making activities during spreadsheet development. Working hypotheses, beginning analyses, insights, and interpretations concerning student reasoning and errors were incorporated. The students' approaches to spreadsheet development were noted (e.g., direct and sequential, hesitant, confused, \) as they were working on the problems. The researcher attempted to determine the types of reasoning

used during the think-aloud sessions and recorded his thinking about the ideas as they unfolded (e.g. trial-and-error, means-end, analogical, case-based, imitation). Also, if errors were made, the researcher noted the errors to discuss during the interview following the think-loud session. The observations included cases where errors were made and then corrected. The researcher recorded observations regarding both effective and ineffective uses of spreadsheets such as copy operations and hard coding values in embedded formulas.

Researcher/Instructor journal

Reactions to class sessions and think-aloud sessions, thoughts and concerns regarding the study, insights, interpretations, beginning analyses, and working hypotheses were recorded in a researcher journal. After each class session the researcher/instructor reviewed notes made while observing the class and added general ideas, trends, and reactions regarding student reasoning and learning associated with error-making. Similarly, after each group of think-aloud sessions and student interviews, the researcher/instructor reviewed the videotapes and notes made during the sessions, recording any overarching themes that emerged from the sessions, as well as modifications that needed to be made in future sessions.

To validate the emerging ideas and concerns and to identify possible biases, a summary of such concepts was reviewed by education professionals. The researcher met with an education professor with an extensive background in education methods and research. The comments and resulting modifications to the study were included in the journal.

Background questionnaire

A survey (Appendix C) was administered in the first session to provide information regarding each students' age, major, background in spreadsheets, computer experience, and mathematics. The questionnaire was be adapted from one previously developed by the researcher/instructor and has been used in numerous computer courses in the past. The questionnaire content was validated by education professionals ensure that it adequately fulfilled the objectives for the research.

Data Collection

Data sources in this research included videotapes of think-aloud sessions, researcher/instructor field notes from observations of students' learning and reasoning during the class and of students dealing with spreadsheet development during think-aloud sessions, interviews with students, student written materials along with mouse/keystroke data for their work on the computer, spreadsheets from the think-aloud sessions, and the background questionnaires. The background questionnaires were completed the first day of class.

The first think-aloud session was held between the third and fourth classes (the third week of the course). Each think-aloud session with students took approximately one hour. The second think-aloud session took place between the fourth and fifth week class periods. The final think-aloud session was held within the week after the course finished. Table 3 indicates the sequence and times allocated.

Table 3.

Data Collection Outline

Week	Session	Activity	Approximate Time (minutes)
1		Background survey	10
		Class observation	120
2		Class observation	120
3		Class observation	120
	1	Introduction and IRB	10
		Practice with think-aloud	15
		Task 1 think-aloud	25
		Interview	15
		Total	60
4		Class observation	120
	2	Task 2 think-aloud	40
		Interview	20
5		Class observation	120
	3	Task 3 think-aloud	40
		Interview	20

During each think-aloud and interview session, students developed a spreadsheet application using the think-aloud protocol. In the session, students narrated their thought processes during spreadsheet development with prompting from the researcher. Participants worked individually and the sessions took place in an environment where the students had access to a computer and the same software used in the course. The interview followed with questions focused on the student's reasoning and errors created during the design and development of the resulting spreadsheet.

In the first session, the participants received an introduction to the study and read the informed consent information. They participated in a short practice activity with the

think-aloud protocol with feedback from the researcher. Most of the time during this session was allocated to the spreadsheet task. Each participant read the instructions and was given a chance to ask questions regarding an interpretation of the task. After beginning the task, the researcher prompted if needed to continue the think-aloud process. If the student stalled on a section, the researcher prompted the student by asking what was causing the stall, asking about the section under construction, asking a question to help to the student make progress, or by offering suggestions. Although an approximate amount of time was allocated for the task, the student was allowed to take as much time as needed, reducing the stress of a time constraint. All participants completed all tasks within a reasonable amount of time (under 30 minutes). After the completion of the development of the spreadsheet, the interview was conducted, focusing on particular segments of the application and reasoning during the spreadsheet development process. The think-aloud session and interview process were repeated for the next two sessions, each in the week following a subsequent class session. The students were requested to withhold sharing information about the tasks or solution approaches with other students in the class.

Data Analysis

The classroom observations were transcribed every week following the class. The field notes included insights, interpretations, beginning analyses, and working hypotheses as described by Patton (2000) about what was happening in the classroom and what it meant. This information was recorded in the researcher/instructor journal. The analysis

was formative at this point to support the continuing direction in gathering ideas about the meaning, causes, and significance of the observations, becoming part of the researcher/instructor field notes. The researcher/instructor identified key elements of reasoning and learning and in particular, those that perhaps led to errors.

The researcher/instructor's personal experience from teaching numerous computer and spreadsheet courses guided the recognition of these elements. To ensure that the researcher/instructor choices were valid and possible biases were identified, these intermediate conclusions were reviewed by other instructors and who had taught similar courses in the recent past. The observations supported the collection of the types of reasoning (i.e. means-end, trial-and error, analogical, imitation) used in the classroom. The classroom observations also indicated concepts that were typically difficult for students and seemed to impact their spreadsheet development accuracy.

During each think-aloud session, the researcher noted reasoning and error-making activities, using the taxonomy described by Rajalingham, et al. (2001) as a guideline. The errors and spreadsheet scenarios involved were added to the error set for each student to be used to determine within-case error patterns as well as in preparation for a comparison of the sets across case studies. Participant spreadsheet errors from classroom activities were also included in the case studies for each student. When the student made the same error during the task that had been made on a previous task and during previous activities, this information was useful in determining not only the reasoning used, but also why the reasoning continued to result in errors. Thus, this information was considered when identifying specific probes for successive interview sessions.

After the completion of all sessions, the videotapes, transcribed researcher/instructor's field notes, written materials and keystroke/mouse recordings of the think-aloud sessions and interviews were analyzed in describing the students' reasoning. Each think-aloud session videotape was partitioned into segments based on an activity associated with a single formula cell or copy operation. Segments associated with the error set were highlighted in the transcriptions. The researcher/instructor selected words and phrases from the transcriptions that seemed to describe the reasoning and activity. Transcribed data from the think-aloud sessions and interviews were then analyzed and coded based on the reasoning identified. The reasoning categories of the taxonomy were used to code the contents of spreadsheet cells. Triangulation from multiple data sources was used to confirm emerging patterns. A segment of the think-aloud for a particular cell in the error set for one student was compared with the interview for the same segment. If the researcher/instructor recorded an observation about this particular error, that information was also integrated into the above data.

For cross-case analysis, the segment data and retrospective reviews of students who made errors in one cell were compared with segment data of others who did not make the errors. The researcher investigated the association between the reasoning and the errors for different students to determine similarly among the students. Emerging associations between reasoning, errors, and sources of reasoning were triangulated comparing videotape data, observations, and spreadsheet data of the students. The data from classroom observations and researchers journal were also used for triangulation.

Table 4.

Identification of the connections among the data sources and research questions.

	Research Question			
	1		**2**	**3**
Reasoning and Learning	**Real-world**	**Mathe-matical**	**Imple-mentation**	**Learning experiences**
Data Sources				
Observations	X	X	X	X
Think-aloud activities	X	X	X	X
Student interviews	X	X	X	X
Researcher/Instructor journal	X	X	X	X
Electronic copy of spreadsheets	X	X	X	

Note: X indicates a connection between a particular data source and research question.

Table 4 depicts the connections among the data sources and their associations with the types of reasoning errors gathered from taxonomy (Rajalingham et al., 2001). The researcher used the information derived from this analysis to develop models for describing the student thinking that led to reasoning errors and how that thinking emerged.

Information obtained from the data analysis was used to answer the research questions.

1. What is student's reasoning that results in *domain knowledge* errors in spreadsheet solutions?

   Domain knowledge errors can be either real-word or mathematical. The think-aloud information, interview, spreadsheet developed, and researcher/instructor's notes were used to determine whether the participant chose the correct algorithm (real-world knowledge). For instance, if the participant chose the correct

algorithm for calculating profit, an error in *real-world* knowledge may have occurred. If the participant chose the correct algorithm for a particular cell, but did not use a correct mathematical formula, there should be a discrepancy between the think-aloud information and the mathematical formula used in that cell. For example, such an error occurred if the participant misplaced parentheses in the mathematical formula. Responses to questions from the interview also contributed to information about the mathematical reasoning. Seeing the error, the researcher directly asked about the subject's translation from the algorithm to formula. The background survey gave additional information about the last mathematics class taken, including how recent it was, and the student's self-rating in algebra, arithmetic, and word problems.

2. What is student's reasoning that results in *implementation* errors in spreadsheet solutions?

   Implementation errors occurred due to lack of understanding of the capabilities of spreadsheets, spreadsheet functions and formulas. Hard-coding, copying, and referencing problems were noted as well as difficulties with spreadsheet functions. An example of this type of error was seen if the participant identified the correct algorithm for profit, and applied the proper formula, but referenced the wrong cell. The participant may have hard-coded specific values in the formula, rather than referencing the cell to access the values for income and expenses. Omitting a parameter from a spreadsheet function reflected lack of understanding of that function. This type of reasoning was evidenced by a disconnect between

the think-aloud information and the formula in the worksheet As the participant indentified the correct algorithm and mathematical representation, but implemented an incorrect spreadsheet represenatation Incorrect use of absolute referencing or fill/copy when the original formula was correct would be an example of such an error.

3. How do learning experiences shape students' reasoning strategies when developing spreadsheet applications?

   The interviews of students were the main source of information for developing answers to the third research question. After each think-aloud session, the researcher interviewed each student regarding errors in the spreadsheets developed during the think-aloud session. The researcher asked probing questions intended to determine the reasoning involved and how that reasoning developed. The reasoning may have been a result of learning that occurred as a result of the formal course, previous learning, a peer learning situation, or student invention. The field notes from classroom observations provided additional information for this question. The researcher observed students developing, acquiring, or demonstrating reasoning that resulted in spreadsheet errors. The researcher/instructor journal provided some information as to common reasoning behaviors for students in this course, validated by other instructors of similar courses.

In the event that a particular error occurred during a think-aloud session, but not during a previous think-aloud session, the researcher/instructor used this

opportunity to investigate the change through the interviews. If an error was made consistently in classroom activities and the previous think-aloud activity, the researcher/instructor tried to determine how the reasoning developed and why the student continued to rely on this reasoning. The participans were asked if they were aware that this approach resulted in an error and if so, why did they continued to use it. Errors and reasoning were sometimes consistent across cases, perhaps indicating a common source for learning reasoning that led to errors.

There may have been an effect on student reasoning from the interviewer's focus on errors. The possibility that the extra time and discussion about particular errors prompted students to modify their reasoning and improved the accuracy of spreadsheet applications they developed. If observed, this modification of reasoning in itself became part of the recorded information. This phenomenon may add to information about methods to improve learning about spreadsheets in order to reduce the error rate.

Finally, the researcher/instructor's observations and background questionnaire from the course added information about possible sources of reasoning developing during the course. The researcher/instructor journal was used in conjunction with the data sources to confirm emerging hypotheses regarding students' development of faulty reasoning.

Chapter IV

Results

This chapter presents the results of the research designed to investigate students' reasoning when creating spreadsheet applications that may ultimately contain errors. The research focuses on the identification of the students' reasoning leading to their errors in the designs of spreadsheet applications. The research questions were framed as follows:

1. What is student's reasoning that results in *domain knowledge* errors in spreadsheet solutions?
2. What is student's reasoning that results in *implementation* errors in spreadsheet solutions?
3. What learning experiences shape students' reasoning strategies that result in either correct or incorrect solution to problems?

In order to provide a basis from which to answer these research questions, the results are organized into three sections: *Student Characteristics and Reasoning*, *Reasoning and Spreadsheet Errors*, and *Learning and Changes in Reasoning*. The section on *Student Characteristics and Reasoning* provides a discussion of each student's background, prior mathematics and spreadsheet knowledge, and basic reasoning approaches while developing a spreadsheet application. Samples are included from the think-aloud sessions and interviews. The next section, *Reasoning and Spreadsheet Errors*, describes the type of reasoning student are relying on when making errors based on the taxonomy developed by Rajalingham et al. (2001) (described in Chapter II). Differences in subject reasoning and spreadsheet application development are identified

in an inter-case study. This section directly addresses the first and second research questions for this study. The third research question is addressed in the section titled *Learning and Changes in Reasoning.* This section describes how student learning in a spreadsheet class impacts error making, elements of the learning process, and changes in reasoning. Excerpts from think-aloud sessions and interviews as well as notes from the researcher's observations are used to provide examples of changes in reasoning and perception that took place during the study.

Throughout this report on the research, the think-aloud sessions are identified by participant designation (A, B, C) and task (1,2,3), or researcher/interviewer (R). That is, excerpts from the think-aloud task 1 for Participant B would be designated by B1, while a designation of A3 would be used for Participant A during think-aloud task 3.

Student Characteristics and Reasoning

The three participants in this study had backgrounds similar to that of the other students in the spreadsheet course in which they were enrolled. Their backgrounds included some computer skills, standard mathematics courses such as college algebra and statistics, and minimal spreadsheet experience.

Participant A

Participant A was a 20 year old sophomore Public Relations major. She was a full-time student who hoped to work in the field after graduation in two years. The spreadsheet course was required for her major. Participant A had successfully taken pre-calculus and statistics. In the survey given the first day of class (Appendix C), she rated

herself as excellent in arithmetic and algebra and good in solving word problems. However, during the study, she experienced many problems with mathematics and repeatedly apologized for not being good in mathematics. This participant rated herself as above-average with spreadsheet skills prior to the course although her actual level was average based on the specific abilities she listed. She liked to use new spreadsheet skills.

Over the duration of the study, as a result of the course, her perceptions of the problems, mathematics, and spreadsheets changed. This subject initially viewed the problems as mathematical but later felt that the spreadsheet was a tool for solving these problems using mathematics. Layout, format, and appearance of the spreadsheet were as important to her as the accuracy of the mathematical results. She knew parts of the spreadsheet were moveable depending on her preferences. As she worked on the problems, she proceeded in a linear manner, pausing when having difficulties that were typically with mathematics. Participant A seemed to perceive the computer as thinking at times as evidenced by her statements like:

> The computer already knows that. The computer figures out how much your profit is going to be.

As insight into the reasoning used during spreadsheet development, excerpts of the think-aloud session from spreadsheet Task 2 provide a basis for further understanding of Participant A's reasoning and thinking. This task involved development of spreadsheet formulas to compute values associated with car gasoline mileage and costs. The starting template (in Figure 4.1) provided the instructions for completion of each component of the spreadsheet. Participants were asked to calculate the amounts for column F (Average

Miles Per Gallon =Avg MPG**)** where for example cell F7 is a function that uses the MPG City and MPG Highway values in the two previous columns.

A2: I'm going to do the *autosum* button and hit average and then I'll move it [the cursor] so only D7 and E7 and then hit enter. And then I'll drag this [F7] down.

But I have an issue ... [Reads error message, interprets it, and concludes that the warning refers to changing the default cell selection of *autosum*]. Oh, that's ok ... [Clicks error away]. And then I'll drag this [F7] down … [Does the fill] … It fills it.

	A	B	C	D	E	F	G	H	I	J	K	L
1												
2												
3							$/gal					
4							2.87					5%
5	Make	Vehicle	tank size	MPG	MPG	Avg MPG	Cost of	EPA	Adj. Cost	Dist.	Cost of a	Cost
6				City	Hiway		going 200 miles	discount		on one tank	tank of gas	with increase
7	Ford	Escape	16.5	19.0	22.0							
8	Ford	Focus	14	26.0	32.0							
9	Subaru	Legacy	16.9	19.0	25.0							
10	Dodge	Stratus	16	22.0	30.0							
11	Honda	Civic	13.2	32.0	37.0							
12	Honda	Hybrid	11.9	47.0	48.0							
13	Subaru	Forestor	15.9	20.0	23.0							
14	Toyota	Forerunner	23	17.0	21.0							
15		Min										
16		Max										
17		Avg										

Figure 4.1 Think-aloud Task 2 spreadsheet starting template.

Participant A used *autosum, a spreadsheet function* introduced in the course, recreating the process learned in class. The use of the *autosum* function was used as a means-end approach to a set of functions - a step towards the goal of solving the problem. This operation was a common source for errors. Participant A viewed the *autosum* function as selecting cells "it thinks" the developer desires in the calculation. In this case the function automatically selects cells C7, D7, and E7 while the problem only calls for finding the average of D7 and E7. Participant A changed the default range selection

demonstrating an understanding of the function for the particular situation. The subject also realized the warning message was not valid for her situation which demonstrated her understanding of the nature of the cell domains for the functions.

Participant A then began working on the next problem in the instructions, calculating the cost of driving 200 miles. The instructions asked that the student determine the cost of driving a 200 mile trip based on average miles per gallon and gas price in cell G2; the resultant was to be inserted into cell G7 (Cost of going 200 miles= 200/Avg MPG * Gas Price).

> A2: So I'm gonna name this [G2] gas price … Oops … [Using a space in a range name, names cell G2 as gas price]. I'll do it in lower case so I don't mess it up [laughs]. And then I could do ... A range [moving mouse around] … I guess. I'm kinda bad at this [laughs]. Then um, I could name this [Selects range F7: F14] average miles per gallon. I don't know if this would work. I don't know if I actually need the average miles per gallon. I'm just going to do: =F7*gas price in cell F7.

At this point, she has made an algorithmic error, omitting the 200 factor from the formula for the solution. The straightforward approach used did not suggest trial-and-error reasoning. Her reasoning appeared based on use of the last item calculated and thus appeared as an analogical reasoning error described by Anderson (1989).

> A2: And then I'm going to fill down to the bottom. Oh geez ... [Moves mouse around, seems concerned but it is not clear if she is concerned over resulting values or decimal point.]
>
> R: So is that the cost of going 200 miles?
>
> A2: Maybe not. Average miles per gallon … Times … Average miles per gallon … I need to multiply by 200 [laughs] … [Starts to retype formula] ... No ... I don't need to multiply it by F7 ... Help!? [her confusion is with the formula].

R: If your car got 20 mpg, and you were going to go 200 miles, how many gallons would you use?

A2: Oh ... Ok [but it did not sound like this had helped her figure it out].

R: How many? You get 20 miles to the gallon.

A2: 20 miles to the gallon. [Repeating instructions using a word problem technique].

R: And you are going to go 200 miles.

A2: 10 gallons. [Quick response]

R: How did you figure that?

A2: I don't know [hesitation]. Divided 200 by 20?

The participant's reasoning process did evolve through a trial-and-error approach and eventually resulted in an impasse. She determined the solution quickly for an easy case, but was not sure how she did that. She did not understand the underlying principles well enough to apply them to the spreadsheet problem. She was able to apply this simple example to the current problem. Her reasoning at this point was not considered analogical reasoning as she was able to do the original problem. Her reasoning here seemed to be case-based reasoning or mimicry.

A2: So I have to do 200 divided by this [F7]. Ok, so 200 ... [Enters formula =200/(F7*gas price) however the correct formula would be =200/F7*gas price] ... Oh, times the gas price, I accidentally put the parentheses on the wrong thing. Miles per gallon divided by … There we go ... [Changes parenthesis, formula is now: =(200/F7)*gas price] … So that's $28.

R: Does that seem realistic?

A2: [Hesitantly] I think so.

R: It is? Remember you said you would have to buy about 10 gallons. And if 10 gallons cost $2.78 a gallon, then you'd spend about $28.

A2: Oh you gave me the formula … My fault [formula had been on sheet] … math and me … Sorry, I'm not the best math student.

This final revision involved an error, although she repaired the error. She may have felt that part of her first attempt was correct, so enclosing it in parentheses would correct the error. The final formula indicated she may not have understood the mathematical order of operations, as the parentheses were not necessary. She later stated that she normally uses too many parentheses.

The next part of the problem used the IF( ) function to calculate a discount based on fuel efficiency of the vehicle. The IF( ) function had been introduced in the previous class session. Referring to Figure 4.1, the students were to calculate the amount of EPA discount in cell H7 (a series of cells in column H contained the EPA discount). Hypothetically, the Environmental Protection Agency (EPA) offered a 5% discount on gas to vehicles with high mileage rates. If the average miles per gallon value was greater than 25 there was a 5% discount on the cost of going 200 miles. Students were instructed to create a formula to calculate the amounts of the discount using an IF( ) function.

A2: We have to do an IF( ) function for this. Remember … =IF [backspaces, then goes to the function tool]. All right, IF, so I hit the IF, if the average miles per gallon is 25, if that's true there's a 5% discount, and then if it's false there's 0% discount. So I can just put 0?

R: Yeah. [She entered =IF(25,5%,0)]

Participant A recalled how to invoke the IF( ) function based on work done in class earlier that week, using analogical or case-based reasoning. The function tool also

was the first place many of the students used when a function was necessary (means-end reasoning). Participant A entered numbers for the arguments of the cells and received a result, but missed the conceptual component where variables were potentially used and logical operations performed. Filling in the parameters with numbers was also basically a means-end reasoning, also, as it achieved the next subgoal of having all parameters provided and producing a result. The result of this formula was 5% regardless of any differences in vehicle gas mileage.

Rather than leave the error for the interview, the researcher/instructor decided to use the opportunity to improve the student's understanding. This situation also provided an opportunity to investigate her reasoning regarding inequalities and translation of English phrases into mathematical representations.

R: See up in the formula bar? See if that does what you want it to do.

A2: If 25 mpg, hmm, I'm gonna have to do less than.

R: It's asking if the average miles per gallon is greater than 25.

A2: Um … [Pause]

R: So where the 25 is, you need to do something else there.

A2: Times maybe? [Pause]

R: Trying to check if it's greater than …

A2: Ok, uh, [pause] multiply? [Pause]

R: Somewhere you have to use the average miles per gallon itself.

A2: [Pause]

R: How do you do greater-than?

A2: Oh yeah … greater than … oh yeah …
[Revises formula to be: =IF(F7>25,5%,0%)].
Oh there, that's where I messed up, I forgot the … Ah … Symbol.

A2: Fill down.

R: Does that make sense?

A2: Yeah, the ones less than 25 are 0.

Based on the researcher's previous teaching of programming courses and mathematics courses, inequality symbols have been consistently troublesome for students. Once Participant A moved beyond that difficulty, she was able to develop a formula that produced a correct percentage based on the mileage of different vehicles. This correction, however, did not solve the problem as stated. The amount of the discount was to be calculated. The error remained and was included in the cost minus discount. In the later interview, this error was corrected. Excerpts from that discussion are included to indicate the disconnect between her fundamental arithmetic skills when used in simple problems and when applied to scenarios.

R: Now, in the EPA discount column, you are to calculate the amount of the discount. You've calculated the percent.

A2: Ok.

R: Calculate the amount and you've got a good start.

A2: If … Then ... Is there a Then?

R No that's already taken care of. You've got the right percentages now. If you want to find 5% of the cost of going 200 miles how would you do it?

A2: The cost … Subtract I guess?

R: If you want to find 5% of 100?

A2: Multiply by 100. So I need to multiply by … [Pause]

R: You have the percent already, so you need to compute the cost.

A2: So I'm going to multiply it by … [Starts to enter cost into an existing formula, but is hesitant as to where to put the change in the formula. She eventually adds it to the right.]

R: Is that right?

A2: Yes, because that [Avg mpg] is under 25 [and the result is correct].

Again, the difficulty was in translating the problem into an arithmetic form. When given a simpler version of the problem, she was able to determine the answer quickly.

In the next column, the cost resulting from the EPA discount was calculated. This determination should be a simple subtraction and the formula was given. The instructions for Adjusted Cost were given as: *In cells I7 enter the formula to calculate adjusted cost for the EPA discount. Cost of driving 200 miles – EPA discount.*. The formula was given in brackets.

A2: [Hesitates] So ... Um … The cost is going to be 200 miles minus the discount ... so ... equals the cost of going 200 miles [types = and selects cell G7] minus [selects cell H7]. Initially, when I looked at the problem, when I knew I was going to be using the 5% I was going to name it. Can I still name it?

Participant A was trying to use a parameter that was not needed in the calculation. She wanted to name cell L4, which contained 5% to be used in a later calculation. The step she completed involved 5% integrated into an IF formula. It was unclear whether the confusion was based on the value being the same. She may not have realized that the value in cell L4 was to be used later.

For the next column Participant A developed the correct formula to calculate the miles for driving on a tank of gas, without the formula being provided. But in an unsure way, she questioned her formulas and the results. The instructions she worked with were: *Distance on one tank: In cell J7 enter a formula to calculate the distance that can be driven on a tank of gas, based on the average mpg and tank size*.

A2: OK, so we're going to need … Column C, tank size, and ... Um … Average miles per gallon, yeah. Distance on one tank … Tank size. Multiply the two? Yeah. So equals, let's see, F7 times tank size … I think … Yes … We'll see … Maybe not.

R: That might be right because you can usually go about 300 miles on a tank of gas.

A2: So the hybrid should be highest … And then the Forester. Yep ok, 416 [miles for hybrid]. Yep cool.

[Instructions, cost of a tank of gas: In cell K7 enter a formula to determine the cost of a tank of gas (formula not given)]

A2: Cost of a tank of gas. So we're going to multiply. We're going to use this gas price [G2]. So gas price times ... Distance on one tank … Oh, no … Wait … Oh, per gallon … Sorry … Tank size … We're going to go back here [C7]. There we go, that's more like it.

For the next calculation, the cost of a tank of gas, the participant made an algorithmic error but corrected it during development. Her original calculation multiplied gas price and distance on one tank, an incorrect calculation. Participant A may have been using the distance on one tank because that was the last value calculated and she thought it must be involved in the next step. This error was an algorithmic error based on means-end reasoning or a faulty analogical process. If she was using the last cell calculated simply because it was the easiest one to grab, it would be means-end reasoning.

However, since the previous steps used the cell most recently calculated, she may have developed an analogical process and thought that a calculation involved the last calculation. This error would be similar to errors observed by Anderson (1989).

The directions for the next column called for calculating the result of a 5% increase in the cost of going 200 miles. The participant developed a correct solution to a part of the problem, the increase. Then she realized it should be the cost after an increase and used new skills learned in the most recent class. The instructions at this point were*: In cell L7 enter the formula to calculate the effect of the % increase in cell L2 on the value in cell K7*.

A2: 5% increase.

R: You can do this any way you want, add columns.

A2: [Rereads instructions] So I just multiply it by 5%?

R: Do you mean 5% or that cell?

A2: The cell [names cell], then I know it will act like an absolute value. So I'm going to do equals increase time the cost of a tank [=increase * K7].

R: So now does the cost of the increase make sense?

A2: Oh, I guess that's just the increase. So I need to … I'm going to use my new skills [excited]. I forgot to add the cost of a tank of gas, include it in the increase [= 1 + increase * cost]. That should make it better.

The formula was wrong, but based on the right idea: *= 1 + percent increase*amount* instead of *=(1 + percent increase)*amount*.

Participant A recalled and implemented a method learned in class. Based on analogical reasoning she developed a solution for this situation, but not a correct one. This type of error seemed to be an arithmetic error in that she developed an incorrect

mathematical formula based on a correct algorithm. The error occurred because she failed to take into account the correct order of operations. In the interview after the session, this section was discussed. The participant had shifted the mathematical approach used since the last task.

R: Does that seem to make more sense than = K7 + increase * K7?

A2: Yes, plus it's a lot less work

R: Now there is only one thing you need to do with it. What this is doing is taking increase as .05. So it's taking .05 plus 1 times … And what was K7?

A2: K7 is cost of a tank of gas.

R: 47.36?

A2: Yes.

R: The way it does operations it's going to do multiplication first?

A2: Right.

R: So you are going to get 1* 47.36, what's that?

A2: That's ... 47.36.

R: And it's going to add .05 to it.

A2: 47.41.

R: You have the right idea, but you need to change it so it added 1 first.

A2: [adds parentheses] I was going to say … It looked really small.

R: Last week it seemed more intuitive to do: K7 + K7*increase … [Writing it down] and this week 1 + increase times amount [(1+increase)*K7]. Does it seem like a different way, or is it better?

A2: It's a better way. I'll probably use it from now on.

R: 'Cause it's less work?

A2: Yeah, and plus I know what the 1 stands for and I have to use parentheses appropriately. Usually I use too many parentheses, actually.

In the last formula, it seemed that Participant A was using analogical reasoning as she was basing a solution on a method covered in class and that had been used several times in activities. Her statement that she knows what the 1 stands for in the equation indicates she understands the underlying principles and is applying them to a new situation.

Participant B

Participant B was a nontraditional male Digital Forensics major in his late twenties with previous community college and military service. He had transferred to the college that year after completing his degree at a community college in a western state. Participant B was a full-time student taking this class as a major requirement. He had taken and passed courses in college algebra and statistics and rated himself as good in arithmetic, algebra, and word problems. His self-assessment prior to the course indicated he had little experience with spreadsheets. While developing spreadsheet applications, he was confident in his mathematics ability. He made efforts to check the numeric results and to assess the accuracy of the work as he developed the solutions; however, he had significant trouble developing the correct mathematical representation of the problems.

As insight into his reasoning during spreadsheet development, excerpts of the think-aloud session from spreadsheet Task 2 provide a basis for further understanding of Participant B's reasoning and thinking, as they did for Participant A's. As mentioned above this task involved development of spreadsheet formulas to compute values associated with car gasoline mileage and costs. The starting template (in Figure 4.1)

provided the instructions for completion of each component of the spreadsheet. Participants were asked to calculate the amounts for column F (Average Miles Per Gallon =Avg MPG**)** where for example cell F7 is a function that uses the Avg MPG used the MPG City and MPG Highway values in the two previous columns.

> B2: Ok, so I can just fill over [selects cells C7, D7, and E7 and uses average choice in *autosum* function] … Average. I can just go like that [prepares to fill down formula]. Wait a minute [all resulting values are the same as cell E7].

Participant B then used the *undo* function and correctly filled down the average. He used *autosum* to accomplish the average function using the approach that was introduced in the course, recreating the process learned in class. The use of the *autosum* function was a means-end approach to a set of functions – a step towards the goal of solving the problem. Participant B used an approach that avoided the possible error by selecting the range first and then invoking the *autosum* average function. His actions did, however, result in an error when trying to copy the formula to cells below using a fill operation. By grabbing the center of the cell to be copied the contents of the cell were copied rather than the formula. This error was a spreadsheet implementation error. The error was corrected during development. He commented on the results:

> B2: Alright, that is my average miles per highway and city which is the number I'm going to base this [E7] on the average of these.

Participant B then moved on to a part of the problem other than the next cell to the right and next instruction. He began working on the calculation for the cost of a tank of gas in cell K7. This method appeared to be means-end reasoning as he was moving on to what seemed the next thing to do.

B2: The cost of a tank of gas would equal ... Is this [cell C7, tank size] times 287 which is a rip-off. Alright so 47 dollars and basically [rounding off] 36 cents for a tank this size. *autofill* ... [Selects cells below K7, rather than K7] ... Oops ... [Then selects K7 and fills down appropriately, but the results are wrong] ... No, that's not going to work.

Participant B has not used absolute addressing, so the formula did not copy correctly. This error was perhaps a reasonable "next effort" that was to be adjusted afterwards. But, when it did not work, he used the *undo* function and began to type the formula in the next cell down.

B2: Doesn't *autofill* work?

The researcher/instructor used this opportunity to try to improve understanding of the concept of absolute reference covered in the previous class. He also he investigated at what point Participant B might associate absolute referencing with the solution to the situation. The researcher/instructor advised Participant B to try the fill down again and to examine the contents of cell H8.

B2: C8 [tank size] times G5 which would be this … [cell below gas price, contains a label]

R: So is there a way of telling it to not adjust a cell?

B2: Um ...

R: Remember absolute reference?

B2: Absolute reference … [Repeats phrase, but without evidence of recalling].

The researcher /instructor led him through the idea of absolute referencing. Clearly he did not recall the technique, but this point was a teachable moment. Participant B's use of fill without absolute reference was an example of a spreadsheet implementation error.

Participant B then surveyed the spreadsheet deciding what to work on next. He began working on the next empty column to the right, calculating the cost of driving 200 miles. The instructions asked that the student determine the cost of driving a 200 mile trip based on average miles per gallon and gas price in cell G2; the resultant was to be inserted into cell G7 (Cost = 200/average mpg * Gas Price).

B2: [He reads instructions and looks at nearby cells for components of the solution]. This [F7, Avg MPG] divided by this [gas price] should be my cost of going 200 miles. Now I want to make sure that's right.

At this point, the participant used a blank cell to check his answer. He multiplied the result of his calculation by 200. But he did not use 200 in the original formula. This error was a real-world error, as he was not forming the correct algorithm to solve the problem. He had used all the components between his original solution and he checked the answer, but not correctly. In checking his answer, he had realized something was wrong and had tried to fix it. The next line of the instructions gave the formula. The researcher suggested that the instructions perhaps provided some hints.

B2: That [F7, Avg MPG] times ... divided by that [G4, gas price] equals my average per 200 miles.

R: Where's the 200?

B2: 200 divided by average miles per gallon. I did this [F7] divided by that [G4]. I did it wrong. [Starts over] 200 divided by the cost, gas price, that's it right there. 200 miles would be $69.68. Absolute reference … [He correctly makes cell G4 an absolute reference and begins the fill down copy feature.].

R: Does your answer make sense?

B2: Average miles per gallon ... Why did I do that? [Starts over]. 200 divided by this [F7], average miles per gallon [initially types formula without leading =, starts over again] ... That did not make sense. That's my answer.

R: It would cost 9 dollars to go 200 miles?

B2: Yep?

R: Does cost of gas enter into that?

B2: Oh, that's what that is [now looking at the formula provided]. Two hundred divided by average miles per gallon times gas price. Twenty eight dollars to go 200 miles. I didn't catch that star [the multiplication symbol in the formula given in the instructions] [begins to enter the formula for the next cell, rather than try fill down]. Now why didn't my absolute reference work when I tried it before?

R: Try it again.

B2: Ok, that's what I did last time and got 28 for all of them.

In his efforts he had tried Avg MPG/gas price, 200/gas price and 200/Avg MPG. At some point, he did look at the formula given, but did not see the last factor. His reasoning was trial-and-error, although he was confident with each attempt. Using his own reasoning would have resulted in a real-world error, as he could not develop a correct algorithm. His reference to not catching the star (multiply) indicated that at one point he was incorrectly implementing an algorithm that was given, which was an arithmetic error. His absolute reference did not give correct results because he used it when the formula constants (=200/gas price). He correctly used the gas price (G4) and absolute reference.

B2: [Entering formula] In G8 … [G7 has already been filled] Now why wouldn't my absolute reference work?

R Try it again … [He gets it].

B2: That's what I did last time and it didn't work, it gave $28 for all of them. [We try it again on G7 and it works. He did a fill-down on the cell instead of

dragging formula. The formula is now: =(200/F7)*gas price] … So that's $28.

The next part of the problem used the IF( ) function to calculate a discount based on fuel efficiency of the vehicle. The IF( ) function had been introduced in the previous class session. Referring to Figure 4.1, the students were to calculate the Environmental Protection Agency (EPA) discount in cell H7 (where column H was the EPA discount). As mentioned above, hypothetically, the EPA offered a 5% discount on gas to vehicles with high mileage rates. If the average miles per gallon was less than 25, there was a 5% discount on the cost of going 200 miles. Students were instructed to create a formula to calculate the amount of the discount using an IF( ) function.

B2: EPA discount with high mileage rates. Ok, this will be fun. Alright … [Opens function tool, and picks IF mutters true/false] ... If [enters F7 >] … Do I have to put equals first?

R: For every formula, but that one already has it.

B2: F7 is greater than 25 … [Formula now is =IF(F7>25)] … 5% discount ... [Silence … Reading information on function wizard ... Then back to instructions ...] Alright if that's greater than 25, I want to multiply .05 [he has entered =IF(F7>25, *.05)] … Going 200 miles = [long silence] …

The current formula resulted in a spreadsheet error, since the spreadsheet program was not able to interpret the formula. This type of error was a spreadsheet syntax error. It was obvious at this point that the participant did not have a good understanding of the IF( ) function. The researcher/instructor used the opportunity to investigate how much conceptual knowledge the student had gained from the previous class and how well the student had integrated that information with the current problem, his algebra skills and other spreadsheet skills. The student needed a great deal of help to develop a correct

solution. Participant B was not sure what should go in this place. At first he wanted to use the value 200, although the problem required the cost of going 200 miles.

R: 5% of what?

B2: 200. [Fast response].

R: 5% of the cost [of going 200 miles] was.

B2: Times G7 [=IF(F7<25, *.05, G7)].

R: Otherwise?

B2: Zero, if false, just zero. [=IF(F7<25, "*.05,G",0)]. Did I type that right?

Participant B included a real-world (incorrect algorithm) and spreadsheet implementation error (by using quotes) in the TRUE parameter of the IF( ) function. This type of reasoning looked like means-end reasoning as he used the function insert tool as a next step and then tried to fill in the parameters with something.

R: The "IF TRUE" needs some work. It needs to be 5% of G7.

B2: Alright, 5% times G7, .05 times G7. [He has=IF(F7<25, ".05*G7", 0]. I don't remember whether we have to do percent or not. I don't think it mattered.

At this juncture, the participant demonstrated analogical reasoning, as he tried to recall how to deal with percents. This knowledge, however, was not completely integrated into his understanding. The problem may have been deeper, as students often have difficulty in arithmetic with the relationship between 5% and .05. The researcher/instructor explained how percents can be used in spreadsheets. Next, the student explored using absolute referencing in the problem, even though it was not necessary for this component of the problem. As absolute referencing was used in the previous component of the

problem, the participant was using analogical reasoning similar to what Anderson (1989) found in algebra problems.

B2: That's under 25, so that's right. [Spreadsheet results show 0. Laughs] Now I need to make this absolute. I don't have to dollar sign everything, do I?

R: Which one do you want to keep the same?

B: I want my formula to stay the same except I want ...

R: When you're looking at Ford Focus … Which cells do you want?

B2: Ford Focuses would be F8. So I want dollar symbol F7?

R: No … In this case you want them to change.

B2: So I want them to change? It's not going to make a difference? [Strange output].

R: There is something in the true condition … Quotes [=IF(F7<25, ".05*G7", 0) so result is .05*G7 rather than computed result]

B2: I don't know why it put those in there; we didn't have them in class.

Again, he used analogical reasoning, recalling the case from class; however, he was also aware that it was not part of his thinking. These errors in this section were spreadsheet implementation errors as they involved improper use of a spreadsheet function that indicated the participant did not understand the parameters of the IF( ) function.

In the next column, the cost resulting from the EPA discount was calculated. This determination was a simple subtraction and the formula was given. The instructions for Adjusted Cost were given as: *In cells I7 enter the formula to calculate Adjusted Cost for the EPA discount. Cost of driving 200 miles – EPA discount.*

B2: [Reading] Enter the formula to calculate Adjusted Cost. [Pause ...]

R: I think it tells you what that formula is.

B2: Adjusted Cost [silence…] 200 miles [types =200-] minus this [I7, cell with EPA discount].

The subject used the value 200 instead of the cost of going 200 miles. Either he associated this problem with the earlier calculation involving 200 or he may have used the 200 because it was mentioned in the problem statement. If he based his solution on the previous problem involving the value 200, he was using faulty analogical reasoning. If however he tried to enter something into the cell based on information in the statement, he may have been using means-end reasoning, entering pieces of the problem rather than understand the problem... The resulting error was a real-world error as he used an incorrect algorithm. The error stemmed from a faulty mathematical representation of the problem, rather than the spreadsheet representation.

R: Ok, but how much does it cost you to go 200 miles? Is that on there anywhere?

B2: [Rereads instructions] Cost of driving 200 miles minus the discount.

R: So where is the cost of going 200 miles? It would normally cost you $28, so what would it cost with the discount?

B2: [Types = clicks on cost – clicks on EPA discount] 28 miles [fills down].

For the next column Participant B developed the correct formula to calculate the miles for driving on a tank of gas, without the formula being provided. The instructions he worked with were: *Distance on one tank: In cell J7 enter a formula to calculate the distance that can be driven on a tank of gas, based on the average miles per gallon and tank size (formula not given).*

B2: **[**Reads instructions. Clicks on Avg Mpg cell. Types = clicks on Avg Mpg F7/ tank size C7]. That would not be right [pause] sorry, I'm tired. Average

> miles per gallon … Actually I think I would want to multiply. [Types =F7*C7] I'll try this: average miles per gallon times tank size. That would be right because my truck gets … And then multiply it because it says to calculate the. [Discussion of various vehicles. Does a fill down. Compares various results].

B2: Is this real data?

R: Yes.

B2: That's cool. [Normally, the next step would be computing the cost of a tank of gas. But Participant B had done this earlier] [Reads directions] I gotta skip K 'cause I did that already. I did that earlier 'cause I just thought it would be a lot easier to start out with.

The directions for the next column called for calculating the result of a 5% increase in the cost of going 200 miles. The instructions at this point were: *In cell L7 enter the formula to calculate the effect of the % increase in cell L2 on the value in cell K7. The solution developed was based on the algorithm: increased cost = original cost * increase percent + original cost*. The participant also added absolute value so fill-down of the formula worked.

Participant C

Participant C was a 20-year-old female majoring in Hotel and Restaurant Management. This student was unable to participate in Task 3, but did complete Tasks 1 and 2. She had taken an accounting course but dropped it due to problems with understanding spreadsheets. She had taken college algebra and rated herself as fair in mathematics but was clearly not confident with her mathematics abilities during the study. College algebra was the only mathematics course required in her major. Participant C typically personified the computer and spreadsheets:

Then… So then *this guy*,..

Now nothing equals *this guy* plus … That … Enter…

Equals *that guy* times *this guy* times … Enter … So 240 cubic feet…

Before *I tell it* to multiply…

She had previous exposure to spreadsheets in the accounting course, but appeared to have learned little from the experience:

R: A lot of times instead of using cells for these formulas, people use the actual numbers … Like 20 x 2x 6…

C1: That's what I did when I first started using … when I first started taking accounting … It was like 'what the hell is this class? Like when I took accounting, I didn't know anything, so that's why I dropped it cause it was like, 'I can't do this', just ridiculous, it was a big loss.

Excerpts of the think-aloud session from spreadsheet Task 2 (Figure 4.1) provide a basis for further understanding of Participant C's reasoning and thinking. Participant C selected all three cells containing numbers to the left of the current cell followed by using the insert toolbar command to insert a function. She then selected average as the function. The result was the average of tank size [C7], MPG City [D7] and MPG Highway [E7]. She then began to work on the second row, row 8.

C2: Insert … Function ... Average ... Do I use all three numbers? [Cells C7, D7, E7] Or just the two? [D8, E8]

R: Two

C2: [Enters formula] I don't need this [C7 is in formula], because I'm so smart. Fill down.

At this point, she realized that fill-down copied the formula, rather than type each one individually. However, she filled down the top cell in the series [F7], which had an incorrect formula in it.

Participant C used *autosum* to compute an average. This situation was the way the function was introduced in the course. She recreated the process learned in class. The use of the *autosum* function was a means-end approach to a set of functions—a step towards the goal of solving the problem. This operation was a common source for errors. The *autosum* function selected cells "it thinks" the developer desired in the calculation. In this case the function automatically selected cells C7, D7, and E7 while the problem only called for finding the average of D7 and E7. Participant C did not change the *autosum* selection resulting in an error. Based on Rajalingham et al. (2001), this error could have been classified as a real-world error as the wrong algorithm was used. However, the error appeared to be due to an incorrect use of the *autosum* range, thus making it a spreadsheet logic error.

Participant C then began working on the next problem in the instructions, calculating the cost of driving 200 miles. The instructions asked that the student determine the cost of driving a 200 mile trip based on Avg MPG and gas price in cell G2; the result was to be inserted into cell G7 (Cost = 200/average mpg * Gas Price).

> C2: [Reads instructions aloud and locates cells to be used]. So price per gallon? 200? so 200 divided by average miles per gallon is 19.2 times gas price... Where do you find gas price? [Looks around spreadsheet] Oh ok, then enter....equals 200 divided by F7 times gas price.

By using the given formula, the participant was able to implement a spreadsheet solution without error. Participant C then moved to the next cell to the right without

filling down the formula. The next part of the problem used the IF( ) function to calculate a discount based on fuel efficiency of the vehicle. The IF( ) function had been introduced in the previous class session. Referring to Figure 4.1, the students were to calculate the amount of The Environmental Protection Agency (EPA) discount in cell H7 (where column H was the EPA discount). Hypothetically, the EPA offered a 5% discount on gas to vehicles with high mileage rates. If the average miles per gallon was greater than 25 there was a 5% discount on the cost of going 200 miles. As described earlier, students were instructed to create a formula to calculate the amount of the discount using an IF( ) function. Participant C initially selected the previous cell, G7, for logical test instead of average miles per gallon, F7. This result was possibly because it was the most recent value computed and she assumed it must thus be used in the next calculation.

C2: So would I just minus equals IF … average miles per gallon, this? [F7] Comma five percent? Do I need another comma with the other percent? Then 0?

Participant C now typed =IF(F7, 5%,0) and was missing the comparison operator in the first parameter of the IF( ) function. The researcher/instructor asked how she represented the part of the problem where the average miles per gallon are over 25.

C2: Ok, [starts typing =F7, then corrects it] equals IF this [F7] is greater than 25, then 5%. [=IF(F7>25, 5%].

It was clear at this point that the student did not have a good understanding of the spreadsheet IF( ) function. The researcher/instructor used this opportunity to explain more about the IF( ) function to help her correct the missing term. The result was an IF( ) function that represented the percent of the discount. This solution was not complete for

this step of the problem, which asked for the amount of the discount, rather than percent; however, she compensated for this data in the next part of the problem. In the next column, the cost resulting from the EPA discount was calculated. This determination was actually a simple subtraction and the formula was given. The instructions for adjusted cost were given as: *In cells I7 enter the formula to calculate adjusted cost for the EPA discount. [Cost of driving 200 miles – EPA discount*].

C2: Do I just do 200 minus [points to EPA discount]? Minus 200 equals ... [Has trouble deciding which cells to use. Eventually decides on = cost – EPA discount percentage times cost.]

For the next column, Participant C developed the correct formula to calculate the miles for driving on one tank of gas, without the formula being provided. The instructions she worked with were: *Distance on one tank: In cell J7 enter a formula to calculate the distance that can be driven on a tank of gas, based on the average mpg and tank size (formula not given)*. Her initial attempt resulted in an incorrect algorithm using division instead of multiplication.

C2: So I do the average miles per gallon divided by the tank size.

R: If the tank size was 15 and the car got an average of 20 mpg, how many miles would it be able to go?

C2: I don't know?

R: Can you figure it out?

C2: Fifteen gallons and 20 miles per gallon? Multiply? 300? [Enters correct formula]

R: Yes

The next step in the application development required computation of the cost of a tank of gas based on data available on the spreadsheet. The formula was not given. The instructions provided were: *Cost of a tank of gas: In cell K7 enter a formula to determine the cost of a tank of gas*.

C2: K7 [reads instructions, also reads instructions for the next column]. So a tank of gas ... The distance on one tank times the price of gas … Sorry the tank size ... $47.35.

R: Does that make sense?

C2: Depending on what you are driving.

The directions for the next column called for calculating the result of a 5% increase in the cost of going 200 miles. Participant C developed a correct solution to a part of the problem, the increase. Then she realized it should be the cost after an increase and used new skills learned in the most recent class. The instructions at this point were: *In cell L7 enter the formula to calculate the effect of the percent increase in cell L2 on the value in cell K7*.

C2: This [K7] times … um ... [mumbling] By the five percent? Multiply five percent times this [K7]. Then do I have to ... Is that it?

R: It should be the total with the increase.

C2: So then add it to ... [Cost of a tank of gas] … So just click on this [current cell] and do plus cost of tank of gas [K7]. Why won't it go there? [Trying with cursor, but eventually types in cell address, K7.]

In the last formula, the percent increase was hard-coded as 5%, rather than a cell reference to cell L2. The result was accurate at this point, but would not be correct if the percent increase in cell L2 were changed. This error was a qualitative error as described in the taxonomy developed by Rajalingham et al. (2001) because it did not produce a

numeric error for the current set of values. The cause of this error could have been faulty analogical reasoning if the student related it to a similar problem in algebra without a variable increase. It could also be means-end reasoning if her sub-goal was to get an answer for this specific case, rather than generalizing.

At this point, Participant C had developed all of the formulas across the top of the spreadsheet. This approach could be used to develop a set of correct formulas across a row and hen fill down all formulas at the same time. . However, she filled down one column at a time. She spent some time assessing the values resuting from the fill-down operation, but it appeared her main goal was to fill-down. The fill-down worked correctly because she used range names. The results were incorrect because of an error in computing the average miles per gallon, which later, many computations were based on.

C2: [Reads instructions describing filling down the columns] All of them?

R: Yes.

C2: [Fills down column F] Do I need to retype this [G] column?

R: Go ahead and see what happens.

It was not clear why she was concerned about this column. It may have been that she realized it had an IF( ) formula and she may have recalled from class that modifications may be necessary, like absolute referencing. She dragged the formula correctly to the cells below.

R: Does it look right?

C2: Yeah, two of them have five percent. Do I need to fix those five percents into something else?

Her concern with the 5% values was notable. She may have thought that just having 5% by itself was not very useful.

R: That should be the amount of the discount rather than the percent of the discount. But let's look at the cells at the top of the column. You accounted for the discount in the next column [=G7-H7*G7 rather than =G7-H7].

C2: So it's ok?

R: Yes.

The participant continued with filling down formulas and then moved on to maximum, minimum, and average at the bottom.

C2: So I want to highlight all of these [C7-C15, includes target] and go to insert maximum?

R: Insert maximum?

C2: Oh, insert function and then maximum. [But the formula is now being entered into D7.] Click D14. But I don't think it's ... it's not in the right cell.

R: Ok, so cancel.

C2: [Enters the formula again with correct target cell.] Then can I just drag it across?

R: Did it pick the right one?

C2: Yes.

C2: For maximum do I do the same thing. Is it better if I highlight the cells before?

R: Yes.

C2: So [moves cursor to C16] C7-C14 right? Not C15 [the cell containing minimum] right?

The instructor took this opportunity to point out a more effective method of selection for these functions. Participant C's use of insert was interesting as most students gravitated to using the *autosum* tool.

Reasoning and Spreadsheet Errors

A spreadsheet error taxonomy developed by Rajalingham, et al. (2001) as described in Chapter II provided categories for classifying and understanding the students' reasoning errors. If students introduced an error into a spreadsheet application due to faulty reasoning, they had developed a way of thinking that was flawed. In the taxonomy, *reasoning errors* were divided into *domain knowledge errors* and *spreadsheet implementation errors*. *Domain knowledge errors* were considered as those errors due to a lack of understanding of the problem in order to accurately model the problem solution. Included in this classification were errors resulting from selection of an incorrect algorithm (*real-world knowledge*) and inaccurate construction of a formula to implement a correctly chosen algorithm (*mathematical representation*). *Spreadsheet implementation errors* were viewed as due to a lack of knowledge of the full use of the functions and capabilities of the spreadsheet package in use including the spreadsheet principles, concepts, constructs, reserved words, and syntax. *Implementation errors* were further divided into *syntax errors* and *logic errors*. Most syntax errors were noted as being caught by the spreadsheet software. *Implementation errors* that existed after development were most likely *logic errors*. The students in this study made each of these types of errors.

These reasoning errors are part of the larger category of *quantitative errors*. Spreadsheets may also contain *qualitative errors*, where future changes may generate errors or the spreadsheet may be difficult to maintain. One of the subcategories of qualitative errors was of interest in this study, hard-coding. In these situations values were used rather than cell references. The result was a solution that may have been accurate for the current case, but may no longer be accurate if input values were changed.

Real-world Knowledge Errors

Spreadsheet errors resulting from the selection of the wrong algorithm were categorized as real-world errors. In making this type of error, the student spreadsheet developer did not know enough about the problem to formulate or select the proper algorithm. Each participant made at least one real-world error. Participant C had difficulties in Task 1, where the problem required that the developers compute the volume of a brick/stone wall based on its dimensions:

C1: The wall would be 20 feet long, 6 feet tall and 2 feet thick, brick costs $2.00 per cubic foot and rock costs $3.00 per cubic foot, but it doesn't say how much rock and how much brick they're going to use.

R: Can you figure out the size of the wall?

C1: So, do I just multiply the dimensions of the wall?

Participants A and B developed spreadsheet solutions that contained real-world knowledge errors in Task 2. They were not able to develop a correct algorithm initially to compute the cost of going 200 miles based on average mpg (already computed) and cost of gas (given) shown in Figure 4.1. Participant B used the last two factors given above in

the formula (average miles per gallon per/cost per gallon), omitting any reference to 200 miles. Then he used a blank spreadsheet cell to check his answer:

> B2: [He reads instructions and looks at nearby cells for components of the solution]. This [F7, Avg MPG] divided by this [gas price] should be my cost of going 200 miles. Now I want to make sure that's right [uses blank spreadsheet cell].

Participant B reviewed wording and stated that this problem asked for the average cost of going 200 miles, interchanging multiplication and division in wording, but sticking with division. The solution process he used is shown in Figure 4.2.

Formula	**Real-life meaning/ Action on formula**	**Result**	**Comments**
=F7/G4	Average Miles per Gallon/ Dollars per Gallon	Incorrect value	"I did it wrong"
=200/G4	200 miles / Dollars per Gallon		"200 divided by the gas price"
=200/G4	Fill down	Same result for all cells	"Why did I do that?"
200/F7		No equal sign	"200 divided by this, Average miles per gallon"
=200/F7	200 miles / miles per gallon	Partially correct, gives gallons	"That's my answer" Researcher/instructor asks if that's right
=200/F7*G4	200 miles/ miles per gallon * dollars per gallon	Correct results	Yeah I didn't catch that star (the multiplication symbol in the instructions
=200/F8*G4	Same formula for next cell	Since earlier attempt at absolute reference did not work.	
=200/F8*G4	Fill down	Works, but starting from the second row	

Figure 4.2 Participant B's solution process

Participant A experienced algorithmic difficulties with the same problem. She initially multiplied average miles per gallon times the gas price, omitting the factor of

200 miles from the problem, like Participant B. Then she tried to average miles per gallon times 200. After determining this action was incorrect, and after prompting questions from the researcher/instructor, she realized the problem required division and divided 200 by the average miles per gallon which was then multiplied by the gas price; the parentheses were incorrectly used to require the division before the multiplication which would have been correct.

In Task 3 a weekly payroll was used in a monthly expense report that was pointed out to Participant A during the interview following the spreadsheet development during the think-aloud session:

> R: Does payroll, $700.00 a month, make sense?
>
> A3: From all the taxes … and everything here, ok … oh, to the month … is that just a week? … [typing] … Ok. So … that's one thing that's tricky for me is remembering … yeah, there's the one thing … with the years … *in class ... when you did the payment thing like this and you had us multiply by 12* it's just remembering the common sense stuff like that…

Such an error might be considered a scaling problem by not accounting for changes in units during the problem (i.e., weeks to months, annual to monthly). Errors such as these occurred several times in the study. In the previous situation, this participant forgot to account for the conversion of a weekly payroll amount to a monthly expense.

In the excerpt below, Participant A referred to a problem done in class where scaling was needed, indicating her analogical reasoning. In the class problem mentioned, monthly payment of a loan, the annual interest was used instead of monthly, but the scaling was more subtle than in the previous problem as it was a component of one of the

terms in a complex function rather than simply a matter of multiplying a result by 4. Her statement, that this multiplication by 12 was something the instructor had them do, was interesting, as if it was something to remember, rather than part of the solution to the problem. In the same task and on the final exam, Participant A had used annual interest rate instead of monthly interest rate for a loan payment problem similar to the one done in class.

In Task 3, the calculation of loan payment was another example of a problem that often resulted in inaccurate results due to lack of real-world knowledge. The students did not fully understand concepts of amortization or compounded interest. Near the end of the course, they were introduced to the spreadsheet function, PMT ( ). This function was useful for calculating a payment due from a loan based on compounded interest. The mathematical, business, and spreadsheet concepts were more complicated than most previous material. The mathematical formula that computed this value was shown to the class as a means of demonstrating what the spreadsheet function was doing and how much easier it was than solving the problem mathematically. Although this material was presented in class, in activities, and homework, most of the students did not have the life experiences necessary to understand this idea. The PMT ( ) function was included in Task 3 and the final exam. The students may have remembered to use the PMT ( ) function, but often did not fully understand what they were doing as they implemented it. In these situations, the student developers used analogical or case-based reasoning as their primary reasoning strategy that was based on prior learning. They also sometimes developed a reasonable, but inaccurate formula, rather than use PMT ( ).

Even after correctly choosing to use the PMT( ) function, errors resulted from the use of improper parameters. The PMT( ) function required that parameters be entered for monthly interest rate, total number of payments (months), and amount. The data for these problems were often given in annual interest rate, number of years of loan, and amount. It was often necessary to scale the interest rate from annual to monthly and to scale the number of payments from years to months.

Participant A had made an error in PMT( ) function parameters, during development. Her result indicated that monthly payment for a $300,000 home at 6.5% APR interest for 30 years would be $19,000. In the subsequent interview, the error was discussed and repaired.

R: Click on that mortgage. See that APR rate of 6.5%? That's an annual rate. For this formula we want monthly rate.

A3: So, I divide it by 12?

R: Right, the B15 … 'cause it was $19,000 … Now it's gone down to $2,000.

A3: And then the profit goes up?

Her reflection that profit was inversely affected by monthly payment demonstrated an understanding of the difference between expenses and income and profit, or possibly she observed and noted a change in profit when the monthly mortgage was adjusted.

Classroom observations by the researcher indicated that students were satisfied getting any number out of this function, and they did not reflect much on the accuracy of the result. When asked during class if they thought their parents were paying $19,000 a month for house payments, they thought that value was much too high. Participant B had similar problems with the same function. His original result for monthly payment was

$7,000,000. In the interview following the think-aloud session, the error was discussed and corrected:

> B3: I think 500 is my monthly payment … [Checks with calculator to see if it is accurate] that would not be correct.
>
> R: The 6.5% needs to be the monthly interest rate … and that's the annual interest rate, the APR there, and it gets compounded every month.
>
> B3: OK, so, I should … Divide this by 12? [Looking at result] it should be more than half, make that more than double ... Yeah, [laughs], 'cause it didn't make sense, 7 million … if I didn't check it, I'd be even less right.

From his experience in class, he remembered that the interest charged on a 30-year loan was almost twice the principal amount. He used this information while repairing the error, but apparently not during spreadsheet development. Interestingly, on the final exam, Participant B did not use the PMT( ) function, but rather developed a formula on his own containing the factors involved. His algorithm was incorrect, although it represented a reasonable interpretation of the problem. Participant A made the same error on her final exam.

Mathematical Reasoning Errors

Mathematical reasoning errors occurred when the student spreadsheet developer chose the correct algorithm, but used an incorrect mathematical representation of it. An example of these types of errors was evident when students used the spreadsheet IF( ) function. During the class and the think-aloud activities, the students generally used the spreadsheet *function insert* tool (Figure 4.3) for functions. This approach was introduced in the textbook for the class. Use of the wizard represented the means-end type of

reasoning as the participants thought of using the *function insert* tool as the next step in the process for using the more complicated functions. Even if they were unsure of what factors to use, this step guided them on a solution path.

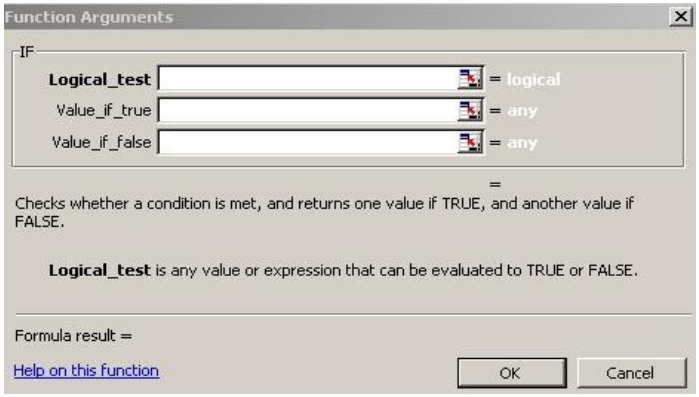


Figure 4.3 Microsoft EXCEL function insert tool

Direct cues for use of the IF( ) function were provided in the instructions for the second task and indirect cues on the third task and final exam. That is, the directions for the Task 2 suggested the use of an IF( ) function for certain portions of the problem. In Task 3, the phrasing of the question was the cue: (i.e. the tax rate is 20% if the employee works over 20 hours, and 0% if they do not).

The IF( ) function required three parameters: a logical test condition, a value or computation for the true condition, and a value or computation for the false condition. As the *function insert* tool provided a framework for the syntax, the reasoning at that point was in response to what the parameters needed to be and how they should be represented. All students in the study had difficulty translating the parts of the problem into correct

mathematical representations for the parameters. In particular, using the correct comparison operators and mathematical expressions were difficult for them. Based on the researcher's experience in teaching both mathematics and computer programming classes students often were confused among the inequality operators (<, >, >=, <=). In this study, such algebraic inequalities were difficult for the students, both conceptually and symbolically. Their difficulties with the higher conceptual level of the functions and their understanding of mathematics from prior experiences translated to difficulties implementing a spreadsheet solution. Students seemed to have a perception that the computer could figure out what was needed if they entered minimal information.

In Task 2, the students were asked to compute a discount on the cost of driving 200 miles based on miles per gallon ratings for different vehicles. The instructions stated that they should use an IF( ) function as this function had just been introduced in class the previous week. Participant B (see Figure 4.4) entered F7 > 25 for the first argument, *0.05 for the second, and nothing for the third [=IF(F7>25, *.05)]. He questioned whether to use 5% instead of .05 then tried to take 5% of 200. The value, 200, was a component of the previous problem, but not part of this one. When that did not work, he figured out that it needed to be 5% of adjusted cost (cell G7). Figure 4.4 shows the actions he used to develop a solution. The researcher/instructor intervened to correct the apparent misconception about quotes in a formula and the formula was corrected (Figure 4.5).

Initial attempt:	=IF(F7>25, *.05)].
Adds information in quotes	=IF(F7>25, ".05,G7")
Then adds 0 as the third parameter:	=IF(F7>25, ".05,G7",0)

Figure 4.4 Evolution of Participant B's solution

	A	B	C	D	E	F	G	H
1	B							
2								
3							$/gal	
4							2.87	
5	Make	Vehicle	tank size	MPG City	MPG Hiw	Avg MPG	Cost of going 200 miles	EPA discount
6								
7	Ford	Escape	16.5	19	22	=AVERAGE(D7:E7)	=200/F7*G4	=IF(F7>25,0.05*G7,0)
8	Ford	Focus	14	26	32	=AVERAGE(D8:E8)	=200/F8*G4	=IF(F8>25,0.05*G8,0)
9	Subaru	Legacy	16.9	19	25	=AVERAGE(D9:E9)	=200/F9*G4	=IF(F9>25,0.05*G9,0)
10	Dodge	Stratus	16	22	30	=AVERAGE(D10:E10	=200/F10*G4	=IF(F10>25,0.05*G10,0)

Figure 4.5 Participant B on Task 2, corrected result

Subject A also had problems with the same IF( ) function. She demonstrated analogical reasoning in trying to remember from class how the IF( ) function was implemented. She initially filled the parameters of the IF( ) function, the last two correctly: =IF(25,5%,0). She then had considerable difficulty determining how to represent the condition: "If the average miles per gallon was less than 25." She had trouble both determining what cell to reference and which mathematical symbol to use. Her first attempt at the function yielded an incorrect result that seemed like the goal was to put something in for the parameters (means-end reasoning). Her mathematical reasoning at this point was unsure, while her spreadsheet reasoning was appropriate. Participant B used trial-and-error within the formula while Participant A got the

arguments of the IF( ) function much more directly using case-based or analogical reasoning, but had difficulty developing the appropriate algorithm.

In Task 3, participants were asked to compute overtime pay for employees in a company. Students understood that when someone worked over forty hours, they received time and a half. Therefore, they exhibited correct real-world reasoning. They demonstrated the ability to figure how much that was for a specific case by breaking it into parts. Reasoning for this problem proceeded most typically as follows:

Mary works 42 hours. She gets $10/hour.

40*10 = $400

2 * 15 == $30

Total = $430

Based on the above reasoning, a generalized algorithm was:

*If Mary works 40 or less hours pay was hours * payrate*

*otherwise pay was 40 * payrate + overtime hours * payrate * 1.5*

with the following spreadsheet implementation:

```
=IF(hours<=40, hours*payrate,
         40*payrate + (hours- 40) *payrate*1.5)
```

Use of the IF( ) function was necessary for the formula to accommodate all cases. Using or developing a formula that extracted the regular hours and overtime hours troubled students, although the student spreadsheet developer could also create columns for regular hours and overtime,

Regular hours: =IF(hours >= 40, 40, hours)

Overtime hours: =IF (hours >=40, hours-40, 0)

The students had completed a similar payroll calculation in a class activity. When they tried to develop a similar computation in the think-aloud session, they were basing a solution process on their idea of time and a half, algorithms like the above, and/or the implementation in class. The algorithm used in class was somewhat more efficient:

regpay = hours * pay

This approach worked no matter whether the person received overtime or not and the "half" of "time and a half" was computed using the formula: =IF(hours > 40, (hours-40)*payrate/2, 0). The part that seemed difficult for the participants was how to express and separate regular hours from overtime hours. This case was evident when Subject A worked on Task 3:

A3: [Starting the problem] They get paid 4 ½ or time and half over 42. Can I use an IF? [Creates new columns]

R: You can make more columns if you want and you can use an IF.

A3: So I'm going to use an IF( ) formula, for this … so …[laughs] … How do I do this? ... IF over 42, so lets see … darn … Let's see … Here we go … IF D4 … is the … all right … if D4 is greater than 40 ... Well, are all the hours paid time and a half?

R: Only over 40.

A3: Just the 2 hours? [muttering] That's not going to work. I thought it would be cool and easy. [Long pause]

R: If you want, you can create additional columns to compute regular pay and overtime pay and use IF.

A3: Um … Oh, I don't want to do that … I want to do it there. So … We'll just go back and do … [laughs] ... All right … this [hours] … Times this [rate] … and then for overtime we get … [pause]

R: Now you can try what you were trying before in another column.

A3: So … IF, go to the formula up … the "IF( ) statement", I don't like to do it without this [function insert] because I can't figure it out [pause]. If D4 is greater than 40 then … then … it's not a percentage though.

This formula was complicated and the researcher/instructor decided the student was struggling and that perhaps this situation was a teachable moment, also providing an opportunity to investigate reasoning during the conversion of a problem into a spreadsheet-based solution.

R: What would the answer be in the first cell, D4? What kind of number are you hoping to get out of there? … For overtime?

A3: I guess … Um … Yes.

R: Well, they're overtime hours, right?

A3: Yes.

R: How many overtime hours did they work?

A3: Two.

R: How'd you get that?

A3: [pause] Oh I see, subtracted.

R: Now tell the machine to do that.

A3: Ok … So … The value is true then … No E … then … D4 …

R: Which cell?

A3: Yeah, D4. A3: … minus 40 … times … 1.5?

R: What's D4 minus 40 going to give you?

A3: Overtime hours. Then multiply it by the value. Could I multiply it by the rate? I probably put that in parentheses. So the parentheses and then … Plus

… I can use 1.5, right? Ok, so times, right? So that's ... Which is right! 'Cause it's times, Cool! Yeah, I look at the number, it says 30, and then ... So that looks alright because 2 hours overtime should be $15.00 and 15 times 2 is 30. And if it's false then 0.

R: And this little box [formula insert] is easier than remembering the formula or trying to figure it out?

A3: Yes, yes.

R: All right.

A3: Well I got to make sure it's … Then I drag it down, Yeah … There's the IF( ) formula [laughs]. Ok, now I wouldn't be able to do it without help ...

R: So it might have been inaccurate?

A3: Yes.

For the same problem, Participant B did not generalize his solution. He did not use an IF() function and often hard-coded numbers instead of using cell references:

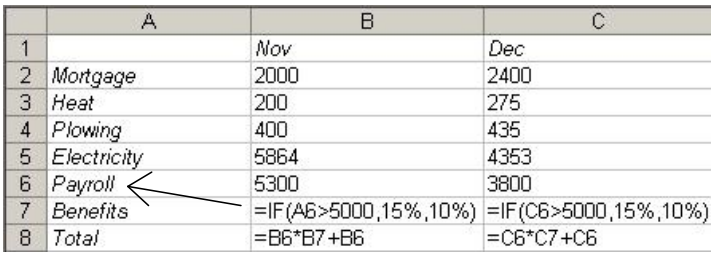
B3: This equals this [rate] times 40 + 30 [this is an error, resulting formula originally entered as = c4*40, corrected =c4*40+30]. 400 plus and extra 15 … Time and a half … Time and a half … Yeah that would be right … 'Cause it would be 10 plus half again what they made, plus that extra would be $15 for each hour over 20. Equals 35 times seven [entered: =D5*C5]. Equals 20 times 65 [entered: =D6 *C6].

F
Gross Pay
=C4*40+30
=E5*C5
=E6*C6
=SUM(F4:F6)

Figure 4.6 Participant B Computation of Gross Pay

Although he spoke using values in cells and entered cell addresses, he used different formulas for specific cases, rather than general formulas to cover all cases.

Participant A demonstrated a mathematical knowledge error on the final exam. On an expense sheet, the benefit amount was to be calculated based on payroll amount in row 6 in the spreadsheet shown in Figure 4.7.



	A	B	C
1		*Nov*	*Dec*
2	*Mortgage*	2000	2400
3	*Heat*	200	275
4	*Plowing*	400	435
5	*Electricity*	5864	4353
6	*Payroll*	5300	3800
7	*Benefits*	=IF(A6>5000,15%,10%)	=IF(C6>5000,15%,10%)
8	*Total*	=B6*B7+B6	=C6*C7+C6

Figure 4.7 Formula view of a portion of Participant A's final exam spreadsheet

Note that the formula in cell B7 of Figure 4.7 calculated a percentage rate for benefit calculation based on the value in cell A6. There were two errors in this formula. The first was that cell A6 was a label, containing no data. The IF( ) function interpreted this case as *TRUE* and always produced a *TRUE* result. This result was the same that would have applied to the value in cell B6 so it did not look like an error. The formula in the next column, C, referenced the correct cell. The second error was one involving the participant's interpretation of the problem. The problem indicated that the student spreadsheet developer needed to create a formula to compute the amount of benefits, rather than the percent. The formula in cell B7 was more appropriately stated as:

=IF (B6>5000, 15%, 10%) * B6

The method used by Participant A did not create a bottom line error as long as the interpretation carried through to the rest of the spreadsheet. In this case, she integrated the data from the benefit computation correctly, but added it to the wrong value. The benefit amount was added to the payroll amount. These expenses were not the total expenses. To get total expenses based on her prior reasoning, the formula was more appropriately:

=B6*B7 + sum(B2:B6)

Participant B made the same error while other students (including Participant C) simply summed all fields, not including the benefit amount.

Spreadsheet Logic Errors

Solving a problem or modeling a scenario in a spreadsheet accurately required understanding the capabilities and programming requirements of spreadsheets. This situation included knowing how to translate mathematical formulas, use built-in mathematical functions, and understanding features that were inherent specifically to the spreadsheet. The last item referred to spreadsheet features such as formula fill and drag, absolute and relative addressing, range naming, and aggregate operations. This category also included spreadsheet functions such as IF( ) and PMT( ), as a student spreadsheet developer needed to gain knowledge for what types of problems where these spreadsheet functions were most likely to be used to solve, as well as implementation issues.

Range naming replaced the need for absolute referencing and seemed to help students understand their spreadsheet. Participant A used range names, formula drag, and absolute referencing although she often asked questions or needed help. Participant B did

not effectively use most of these features. Participant C saw the power of the features and tried to use them effectively, but had more problems than Participant A. Participants A and C were as concerned with the layout and appearance of the spreadsheets developed as they were with the accuracy of the spreadsheet. They moved sections of the spreadsheet around to make it look the way they wanted.

Formula fill and drag operations were features of spreadsheets that provided capabilities for copying formulas to adjacent cells. From the classroom observations and the researcher/instructor's background from teaching spreadsheet courses, these features are often difficult to learn and implement correctly. While students often have become comfortable with copy operations, there are occasions when only certain elements of a formula should vary. In these cases, absolute referencing is necessary. This type of cell reference was more difficult for the students to understand. Another method of keeping an element of a formula static was to create a range name. This strategy also improved the readability of the spreadsheet application. Portions of the researcher's notes and observations regarding spreadsheet implementation provided insights into the reasoning used:

> Participant B in Task 2: Now correctly uses absolute reference on cost of gas and fills down, getting the same answer for all (200/2.87) as interviewer asks if it makes sense. After a pause, the participant asks why his absolute reference didn't work. He repeated the procedure and this time it worked. He was concerned that when he did it before, it made all the numbers the same. He had used the fill cell techniques instead of the drag formula.

At this point, Participant B tried to use an absolute reference resulting in creating the same values as he used the *fill down* spreadsheet feature.

R: Does that make sense?

B2: Why did I do that? That's not right.

R: So it would cost $9 to go 200 miles? Where is the 200 in there? [Student pauses to think]

B2: 200 … I did this divided by that, I did it wrong. 200 divided by the gas cost ... That's it right there.

Having a correct solution for this cell, the participant tried to copy the formula to other cells. He should have been able to drag the formula from cell C7, but had problems. When the formula-drag technique did not work, he began to enter new formulas for each cell in column G, rather than using copy operations.

B2: Now why wouldn't my absolute reference work?

R: Try it again … [This time the formula-drag is successful].

B2: That's what I did last time and it didn't work. It gave 28 for all of them.

The participant tried formula-drag again on the same cell and the results were correct. Apparently, he had filled down on the cell instead of dragging the formula, so the value in the cell was copied rather than the formula. In this example, he used trial and error in the mathematical representation of the problem and in the spreadsheet technique of absolute referencing of cells. Without intervention, this approach most likely would have been inaccurate.

Participant C made an error in Task 2 (Figure 4.8) that occurred due to improper use of spreadsheet functions. The *autosum* function automatically set up the sum formula based on what the spreadsheet interpreted as reasonable. This situation usually meant the cells above the current cell or to the left of the current cell were parameters. In the

spreadsheet segment shown in Figure 4.8, the subject was supposed to find the average of the in-town mileage rating and highway mileage rating. But instead, she averaged the preceding three columns, which included tank size in the computation.

	A	B	C	D	E	F
1						
2						
3						
4						
5	**Make**	**Vehicle**	**tank size**	**MPG City**	**MPG Hiway**	**Avg MPG**
6						
7	**Ford**	**Escape**	**16.5**	**19**	**22**	=AVERAGE(C7:E7)
8	**Ford**	**Focus**	**14**	**26**	**32**	=AVERAGE(C8:E8)
9	**Subaru**	**Legacy**	**16.9**	**19**	**25**	=AVERAGE(C9:E9)
10	**Dodge**	**Stratus**	**16**	**22**	**30**	=AVERAGE(C10:E10)
11	**Honda**	**Civic**	**13.2**	**32**	**37**	=AVERAGE(C11:E11)
12	**Honda**	**Hybrid**	**11.9**	**47**	**48**	=AVERAGE(C12:E12)
13	**Subaru**	**Forestor**	**15.9**	**20**	**23**	=AVERAGE(C13:E13)
14	**Toyota**	**Forerunner**	**23**	**17**	**21**	=AVERAGE(C14:E14)

Figure 4.8 Participant C's error on Task 2

Hard-coding and Inflexibility

Hard-coding of values rather than use of cell references was a common practice and source of qualitative errors in novice spreadsheet developers. As they initially learned how to enter formulas into spreadsheets, calculator-type reasoning might be used to solve a single instance of a single problem. This calculator-type reasoning revealed a user entering numbers to obtain a result. If the values needed to be recomputed later, the numbers were reentered. If a different set of values needed to be computed, they were entered with the same operators as the first set.

As a qualitative error, this kind of reasoning had the capability to produce correct results for a specific instance, but provided no flexibility for an application that might be used for a diverse set of inputs. As the students progressed in learning to develop spreadsheet applications, hopefully, cells were perceived as variables and formulas as programming segments using those variables. As evidence that hard-coding was common amongst novice spreadsheet developers, on the final exam, 39% of the spreadsheets contained some form of hard-coding.

Another goal in programming was to develop a general formula that handled all cases. In spreadsheet development, novice student spreadsheet developers sometimes used separate formulas for each case of a series, rather than developing a flexible formula to express the correct result based on input values, as cell references. Participant B did not generalize often and used quite a bit of hard-coding throughout the three tasks in the study. In Task 1, Participant B did several calculations using numbers that did not change if the problem parameters changed. In the Task 3 payroll calculation shown in Figure 4.9, components of the formula for Gross Pay were hard-coded for the first entry, while the other cells used a formula.

C	D	E	F	G	H	I
				0.08	0.2	
Rate	**Overtime Pay**	**Hours**	**Gross Pay**	**Fed tax**	**State Tax**	**Net pay**
10	2	42	=C4*40+30	=G1*F4	=H1*F4	=F4-G4-H4
7	0	35	=E5*C5	=G1*F5	0	=F5-G5
6.5	0	20	=E6*C6	=G1*F6	0	=F6-G6
	=SUM(D4:D6)	=SUM(E4:E6)	=SUM(F4:F6)	=SUM(G4:G6)	=SUM(H4:H6)	=SUM(I4:I6)

Figure 4.9 Participant B hard-coding in Task 3

The first entry was the only one to qualify for overtime pay and Participant B developed a specific formula for that instance. As can be seen from the spreadsheet segment shown, the cells in other columns were computed with different formulas depending on the case. In addition, the formula for State Tax required an IF( ) function to accommodate different conditions. The result for the second employee was inaccurate as that employee worked over 20 hours and therefore the state tax should have been non-zero. Likewise, a general formula for Net Pay was needed. The first entry in that column was correct and should have been filled down for all cells.

The numerical results for this spreadsheet were accurate for this set of inputs, except for the error in State Tax for the second employee. If the second or third employee worked over 40 hours one week, the results for overtime pay were not accurate. As the data size for this application was small, this approach may have seemed reasonable. A larger data set may have forced a decision between typing in individual formulas and generalization.

Student Perception of Error-Making in Spreadsheets

Interestingly, students perceived spreadsheet errors and error-making in an unexpected manner. The mental penalty for making an error did not appear to register as deeply as making errors in a mathematics class. Participants A and C tended not to have much confidence in the accuracy of the spreadsheet application they developed. Participant B, however, did think the spreadsheets he developed were accurate. All three participants seemed to feel errors in spreadsheets were easy to repair. The following

segment provides the interview with Participant A following the third task, in which the interviewer has pointed out an error in the profit calculation.

> A3: So I have to *fix* the net profit. Oh, income combined with expenses … oops, there. I guess it's just remembering that you have to do this. *But there you go, it's easy to fix***.**

> A3: But everything changes with this … And I think this is really … It's my favorite thing about this [spreadsheets] is that … You can mess up but it doesn't even really matter … Because you can then just go back …

The same participant reflected on error-making during Task 1:

> A1: Usually when I first do something, I usually make mistakes, but then I remember it and don't make it again.

Participant C felt comfortable developing a section of the spreadsheet knowing she could move it later. In this case, a numerical error was not the result. She felt the position and formatting of this particular section might need to be changed later.

> C1: I'm doing something wrong but I know how to do it so I'll just do it first and you can move it around if you need to.

Learning and Changes in Reasoning

The accuracy of a spreadsheet application should be related to how the student spreadsheet developer learned to reason with spreadsheets as a tool. The learning process and changes in reasoning may reveal effective teaching and learning processes in spreadsheet education. This information may provide insight into how end-users learn to reason accurately during spreadsheet development. The next sub-sections examine changes in spreadsheet development by students during the study and changes in perceptions regarding spreadsheet application development.

Changes In Spreadsheet Development During The Course

In Task 2, Participant A exhibited difficulty with the IF( ) function and on representing a problem in the parameters of the function, as well as with mathematical notation, such as the 'greater than' (>) symbol. Task 3 provided an example a week later. In the meantime, another class session exposed students to the IF( ) function. Part of Task 3 required participants to develop a formula to compute an amount deducted from an employee paycheck based on number of hours worked. This task required an IF formula similar to: *=IF (hours > 20, 10%, 0)*gross pay*. Participant A did not have the problem with the 'greater than' symbol as in the previous task. She also was able to integrate the IF( ) function into a larger expression with little trouble.

A3: State tax is if they work a certain number of hours. So if the employee work over 20 hours a week they pay 10% of the gross pay no other deductions, less than 20 hours a week, no deductions, Ok. So … We're going to go to … does that include 20 hours a week? They worked over 20?

R: Right. It doesn't include …

A3: 20. So it would include an IF statement … So … I'll just go to the function and then bring this up again and … if … Let's see … The hours are over here … Are greater than 20 ... then … They're … So does the gross pay come from their … Oh, I'm sorry. They pay the 10% of the gross pay. OK. So it's 10% times the gross pay?

R: Yes.

A3: Ok. So they pay ten percent times .1 … Then 0 … [typing] That works out because Mary's 20 hours and she doesn't have to pay state tax.

Participant B's knowledge and approaches to solving problems did not change much during the study. He quickly developed formulas which were quite often wrong. He used hard-coding and frequently reverted to entering individual formulas when a more

general formula could have been used with the spreadsheet fill operation. He was the most confident of the three participants and most confident in the accuracy of his resulting spreadsheets.

Participant C's spreadsheet ability improved during the study. She entered functions with direction and prompting, but was not proficient at developing original algorithms or solving problems without a template or directions.

The other notable conceptual change was the representation of a percent increase. Modeling a potential increase or decrease in a value in a spreadsheet was a common experience to compute the "what-if" value in another cell. The desired increase was stored as a percent in a cell as shown in cell B2 in Figure 4.10.

	A	B
1		
2	% Increase	5%
3		
4	Amount	What-if
5	30000	31500

Figure 4.10 Percent increase example

A common way to approach this problem was to calculate the amount of the increase and add it to the initial amount:

amount_of_increase = amount * percent

amount_with_increase = amount + amount_of_increase

The spreadsheet formula in cell B5 then became: = A5 + A5*B2

Algebraic factoring provides a more efficient formula, but at a higher conceptual level:

amount_with_increase = amount * (1+increase)

The spreadsheet formula in cell B5 would be: = A5 * (1+B2). Participant A experienced this conceptual transition during the weeks of the study. In Task 2, she needed to represent a 20% increase in a value. She created the formula =(C9+G12)*0.2+(C9+G12) even though (C9+G12) had been computed in another cell. The following excerpt from the interview followed the think-aloud session for Task 2.

A2: That might work but it doesn't seem right.

R**:** You could use the cell where the multiplication has already been done. Also, for the increase, you could use one plus the percent increase times the value [writes it out on paper].

A: No

R: So total * 20% and then total plus markup 3 different cells seems better?

A: Yes?

Participant A had a similar problem in Task 3

R: [Explaining problem with percent increase, which had been done as amount *1+ Percent_increase instead of amount * (1 + percent_increase)]. It's going to do the multiplication first and then add .05. You have the right idea, but you need to change it so it added 1 first.

A3: I was going to say … It looked like a really small...

R: This is a change since last week. You did it as K7 + K7*increase and this week as (1+increase)*K7 Does it seem like a different way, or is it better?

A3: It's a better way. I'll probably use it from now on.

R: 'Cause it's less work.

A3: Yeah, and plus … and I have to use parenthesis appropriately. Usually I use too many parentheses

The practice task (Figure 4.11) provided a surprise example of error-making and a measure of change through the five week course. The practice task asked students to find the total amount of money resulting from different amounts of coins. All three students entered formulas to compute the total number of coins, but believed they were computing the total amount of money. It was not until they were questioned that they realized they did not really answer the problem.

Coins	Pennies	Nickels	Dimes	Quarters	Totals
Lauren	2	4	2	2	
Kristen	3	2	1	2	
Ryan	4	4	2	3	
Grand Total	**9**				

Figure 4.11 Practice Task Errors

A real-world error implied the developer selected an incorrect algorithm and that was true in this case, possibly due to means-end reasoning. Having recently learned *autosum* and seeing columns of numbers, use of *autosum* may have seemed like the next correct step to take. The problem may have reminded the participants of one from an earlier class activity involving *autosum*. Since they knew how to compute the answer upon reflection, the cause of the error may have resulted from something other than not understanding how to select the appropriate algorithm. If the error was due to faulty mathematical reasoning then the students incorrectly implemented a correctly chosen algorithm. The participants did have some problems understanding how to create a

formula to represent the problem, seemingly due to needing to include the amount each coin was worth. The participants may have expected all data to be resident on the spreadsheet. But once they understood the idea of including coin amounts, their developments of formulas seemed to proceed well.

Alternatively, this error in the practice task may have been due to spreadsheet implementation logic. The participants could have gotten into trouble by using *autosum* (means–end reasoning) without thinking. Having learned *autosum* and fill, they happily applied it to the problem and were happy with the results.

Based on this phenomenon, a similar problem was included on the final exam. The error was not duplicated. The problem appeared more complex so students may have paid more attention to it. They also had two weeks more experience with spreadsheets by the time they took the final exam.

Student Learning From the Course

From Participant A, Task 3:

A3: I wouldn't have been able to change … You just can't change stuff plus I wouldn't be able to do, like, the two sheet thing. I wouldn't be able to understand how you could … Well I would understand how you get this but I would probably just put it, put the values instead of like …

R: … Like using a reference?

A3: You can change everything and it will automatically change.

A1: I probably could not have done this before the class. I could have done the sum thing and graphs - I'm good at graphs.

B3: …Because of class … If I hadn't taken that class I wouldn't have a clue what to do…

Learning Difficulties

Part of the interview process for most of the tasks asked the students what was difficult about what they had just done. From Participant C Task 1:

R: What is the hard part about the formulas in this? Is it the mathematics thinking ... Is it more like a word problem or something to do with the spreadsheet?

C1: I think it's more like the formulas you have to use. I've never been sure … Of what formulas … I know, like … I have an idea of like what you have to do. Like, for the percent and stuff the percents, like people use the decimal …

From the interview with Participant A after the final think-aloud task.

R: So you were ok with all the formulas?

A3: Yes

R: Are some harder than others*?*

A3: I'd say I'm getting used to all of them. So…maybe…just the mathematics and figuring out just which equation to use. Just the simple mathematics.

Yet for Participant B during the interview after the Task 3, this discussion occurred.

R: Were any of these spreadsheet concepts harder to learn in the beginning than any of the others?

B3: … Just the more complex formulas.

Perception Change

What about students's perceptions as they progressed through the course? Two notable perceptual changes were examined. One of the changes was a transition of the students' perception of the problem and the spreadsheet as a problem-solving tool. The following excepts are from the interviews following Tasks 2 and 3:

R: So when you're thinking of this is it a spreadsheet problem or a mathematics problem?

A2: I think excel helps with the mathematics and applications. I still think of it as a mathematics problems, but it's easier with a spreadsheet.

And from Task 3, this exchange was noted:

R: Are some formulas harder than others?

A3: I'd say I'm getting used to all of them … So maybe just the mathematics and figuring out which equation to use ... Just the simple mathematics.

R: Was it because you may have misinterpreted the problem, had trouble with the mathematics or made a spreadsheet error*?*

A3***:*** I think you need both. I see it as a spreadsheet problem… But before (the class) it was mathematics. I think it's applying your mathematics skills but using a spreadsheet to help you do it**.**

Yet, Participant B expanded as follows:

R: When you think of this now do you think of this as a math thing or a spreadsheet thing?

B3: Uh, spreadsheet, because it's easier…

R: Did that change throughout the course?

B3: Yeah, like when I started it, I rarely used the spreadsheet for anything … I didn't really know what good it would be … And after doing it … This is easier because you can do it faster and a lot of calculations quicker … Than sitting there with a calculator.

Participant C's perception of the spreadsheet as a problem-solving tool changed during the study. She began to view the spreadsheet as a tool that could build applications for problems.

Another interesting observation was that the concept of computing percent increase in this fashion seemed to have become a spreadsheet technique, rather than the

result of factoring from algebra. This phenomenon was observed several times during the class and study. It seemed to represent a paradigm shift as students substituted a spreadsheet approach for their problem solving rather than their pre-existing mathematical approaches. This shift itself was observed to be a source of errors.

The participants appeared to have changed their concepts of summary statistics (sum, average, min, max) after some experience with spreadsheets. These functions seemed to become spreadsheet functions or a button to hit (*autosum*, with options) without an association with the underlying mathematical concept. Students skipped the real-world reasoning and mathematical reasoning and quickly initiated a spreadsheet solution. In a means-end approach, a spreadsheet process replaced the conceptualization of simple statistics. Rather than selecting an appropriate algorithm and developing a correct mathematical representation of that algorithm, a spreadsheet procedure or function was selected.

The students seemed to feel there was only one mathematics solution to a problem, whether they knew what it was or not. However, they were able to implement different versions of a spreadsheet implementation. If they were unable to remember how to use absolute referencing or fill operations, they were not hesitant to enter formulas for each cell. If they did not remember how the SUM function worked, they added the values by cell. They demonstrated problem-solving skills within the spreadsheet development by selecting approaches they knew how to work from a possible set of approaches, often within a series of cells that should have the same formula.

The participants seemed to proceed in a linear manner when working with the spreadsheet and spreadsheet functions. They seemed to view the process as filling in the cells like a puzzle. They slowed down or stalled during algorithm development and their mathematical phases were often guessing, trial and error, or a means-end reasoning based around obtaining a value to be displayed in a cell. In the spreadsheet component, they tried multiple methods, particularly if one was not working. In their mathematical reasoning, they seemed to think there was only way to solve the problem, and they did not know that way. The participants began to see the spreadsheet as a tool to solve mathematical based problems. Following are participant responses to the question *"* When you think of this problem, do you think of this as a math problem or a spreadsheet problem?

> Participant A after Task 2: *"*Excel helps with the math problem. It helps me do that with applications, um…but I still think of it as, this was definitely a math problem, figuring all this out."
>
> Participant A after Task 3: "I think you need both. I see it as a spreadsheet problem…but before (the class) it was math. I think it's applying your math skills but using a spreadsheet to help you do it."
>
> Participant B after Task 3*:* "Spreadsheet, because it's easier. When I started it, I rarely used the spreadsheet for anything. I didn't really know what good it would be. After doing it, this is easier because you can do it faster and a lot of calculations quicker than sitting there with a calculator."

Summary

The objectives of this research were to investigate students' thinking that resulted in reasoning errors during development of spreadsheet applications and how their

learning experiences affected the reasoning strategies they used. Three volunteers took part in think-aloud protocols while developing spreadsheet applications. While performing the tasks involved in this study, students made a variety of reasoning errors. In fact, each participant made errors in both the major categories of reasoning errors: domain-knowledge errors and spreadsheet logic errors. These results seemed typical of the students in the class used for this study. On the final exam, almost all student spreadsheets had numerical, bottom-line errors. As with the think-aloud tasks, the spreadsheet errors included domain-knowledge errors, mathematical reasoning errors, and spreadsheet implementation logic errors.

This study also investigated the learning experiences that shaped students' reasoning strategies in developing spreadsheets. Each participant's learning experience included his or her mathematical learning and knowledge prior to the course, coupled with previous spreadsheet knowledge, and the new knowledge integrated from the course. While there was a noticeable increase in spreadsheet knowledge during the term, the mathematical knowledge remained based on previous mathematics courses, although this course may have served to remind them of some mathematical principles they had not used recently.

The participants' learning experiences in the class influenced the spreadsheet reasoning used to solve problems as they integrated new information into existing knowledge. Each new element of spreadsheet skills and knowledge learned provided additional opportunities for error making. The more experience students got with existing spreadsheet skills and knowledge, the better they were able to use effective reasoning to

create spreadsheet applications that were accurate. In general, participants in the study and students in the class seemed much surer of themselves and the methods to use with the spreadsheet component of application development than the algorithmic and mathematical component.

Although the purpose of the taxonomy developed by Rajalingham et al. (2000) was to classify errors, reasoning used sometimes fell into multiple categories. For instance, if student spreadsheet developers did not understand amortization, they lacked the knowledge to model it resulting in *a real-world knowledge error*. In addition, if they were unaware of how to use the amortization function in a spreadsheet correctly, they may have made a *logic error*. Even if the student spreadsheet developers understood amortization and the associated spreadsheet function, they may have made *mathematical errors* in the function arguments. Another example of an error classified into multiple categories was the improper implementation of the spreadsheet *autosum* function. When possible, the classification was made based upon the specific instance.

Participants also used multiple reasoning approaches within a single problem. Often, students used the function insert tool to start the solution process (means-end reasoning), selected the appropriate spreadsheet function based on classroom activities (analogical reasoning), and then had difficulties with the mathematical representation of the algorithm they selected (trial-and-error reasoning).

The reasoning that participants used involving domain knowledge included how they thought during algorithm development and mathematical implementation of those algorithms. One goal of this study was to determine the reasoning used by students that

resulted in real world errors (algorithm development) or mathematical errors. The students in the study used a variety of types of reasoning during the algorithm development and mathematical implementation phases when creating spreadsheet applications associated with the research tasks, including trial and error, guessing, means-end, analogical, giving up, and asking for help.

The following sections summarize the results in regards to the research questions. The first research question dealt with students' reasoning that resulted in domain knowledge errors and the second with students' reasoning that led to spreadsheet logic errors. The third question focused on learning that may have affected reasoning associated with correct and incorrect spreadsheet representations of a problem.

Research Question One

Research question 1 considered: What is student's reasoning that results in *domain knowledge* errors in spreadsheet solutions? The two subcategories of domain knowledge errors are summarized in conjunction with the learning that took place that may lead to incorrect reasoning in these areas.

Real-world Knowledge

The participants exhibited problems with real-world knowledge in determining the amount of material needed to build a wall (Task 1), gas mileage calculations (Task 2), payroll deductions, payroll overtime calculations, and amortization problems (Task 3). In several of these cases, the students lacked the life experience that contributed to understanding these situations. In others, the difficulty was developing and algorithmic

representation of the problem. In Task 1, Participant C had difficulty thinking about how to find how much material was needed for a wall, given the dimensions. Her reasoning process encountered two hurdles. She computed the volume as a middle step (not intuitive for her) and she stopped trying to solve the problem. After the researcher suggested finding the volume of the wall, she was also not sure of how to compute it. She then made what sounded like a guess at the algorithm, but this result was probably based on previous real-world knowledge on how to compute volume.

Inaccurate spreadsheets resulted from developers being unaware of specialized formulas or algorithms, such as amortization. In Task 3 of this study, the participants were guided through the instructions to use the PMT function to compute monthly payment on a loan. It was clear during classroom observations and the interviews after the think-aloud tasks, that most students did not understand why they were using the PMT function, although it had been explained in the class. On the final exam there was no direct cue to use the PMT function to compute monthly mortgage payment. Several students, including one of the study participants did not use the function, but developed a reasonable, yet inaccurate, formula. The real-world information about amortization and/or the spreadsheet information about the PMT function did not become integrated into their existing knowledge. Task 2 involved a series of computations associated with gas mileage. For many of these computations, the students in the study had difficulties developing correct algorithms. They tried two-factor solutions when the correct solution required three factors. In these cases, the reasoning degraded to trial and error combinations of pairs of factors and multiplication and division. There was sometimes

confusion over whether to use multiplication or division. Errors resulted due to improper use of parentheses and not accounting for the mathematical order of operations. The participants did not use units as a guide to the solution and their reasoning process seemed similar to students in mathematics courses solving word problems. Mathematics settings, however, usually suggest structure and/or process for the composition of an algorithm from a problem.

There were a number of scaling errors, which were usually addressed during the interview following a think-aloud sessions. This type of error was due either to misunderstanding the problem or an omission. In the former case, the participant was unaware that some sort of conversion was needed from one time unit to another. In the latter situation, the participant may have focused the reasoning on larger aspects of the problem and omitted the unit conversion.

Mathematical Reasoning

The reasoning that students used in developing mathematical representations of the algorithms chosen was either straightforward or affected by a difficulty in transfer of prior mathematics skills and knowledge to the current situation. Both approaches resulted in correct and incorrect results. In a straightforward approach, participants progressed confidently but sometimes made an error without realizing the error. In this situation, they seemed to be basing their solution on previous knowledge. If an error resulted, that knowledge or the transfer process was faulty. This type of error was particularly evident with Participant B. It was more common, however, for the participants to question the

mathematics involved such as when trying to decide whether to multiply or divide, using and representing percentage, or selecting the proper comparison operator.

Common mathematical reasoning errors described by Rajalingham et al. (2000) included difficulties with percentages, improper use of parentheses, and not understanding order of operations. Participant A exhibited difficulties with the latter two. She also had difficulty determining which comparison operator to use in Task 2 when using the spreadsheet IF( ) function. At some point in the study, all participants questioned their use of percents and how to represent them.

These errors in reasoning suggested that the students did not fully understand these mathematical concepts and were not able to transfer their understanding from previous mathematics classes to a new situation. Analogical reasoning was not a reasoning method that supported them. The situation might have been different enough from their original mathematics learning environment that transfer did not occur. The mathematical components of Task 1 and Task 2 were different from those in class activities, reducing the possibility of direct transfer. Task 3 was similar to an activity done in class and to part of the final exam so there may have been some more direct transfer of mathematical reasoning, although there were mathematical errors in both Task 3 and the final exam.

Participants also at times tried to use a factor from one problem in subsequent problems, even though that factor was not part in the problem. This event could be an error due to analogical reasoning based on the previous problems.

Research Question Two

The second research question was: What is student's reasoning that results in *implementation* errors in spreadsheet solutions? Several spreadsheet operations might be intuitive after they were learned and mastered, but the students were novices when they were presented in the study. The participants had some previous experience with spreadsheets, but not to the extent covered in this introductory course. Their previous knowledge may have been more at the level of a spreadsheet application user than of a student spreadsheet developer. If the participants did find a value out of a function, they appeared to be happy, even in cases when the results were unrealistic. There was no critical evaluation of the results. This type of reasoning appeared to be means-end reasoning with the goal to fill in the cell. In fact, this type of reasoning seemed true for most values generated in their solutions.

Cell referencing was one of the basic techniques that involved new thinking on the part of students. The concept was akin to use of variables in algebra or programming rather than constants. Use of referencing increased the flexibility and generalizability of the spreadsheet applications. Most students in the course learned this technique easily and by the end of the course they were more likely to use it intuitively. Participant B had difficulty and occasionally used a number instead of a cell reference. He also did intermediate calculations in his head and entered the results, rather than use cells to hold values. His formulas were often correct for the current data set, but were inaccurate if input values were changed. His focus appeared to be on the localized view of the problem rather than developing a flexible application that could be used for different sets

of data. He did not improve with new knowledge from the course, but rather integrated the same non-efficient skill set into spreadsheets that were more complex. The course materials and activities had prescribed developing generalized and flexible applications. This participant did not seem to understand that intent.

The copy and fill operations reduced the amount of typing necessary to complete a spreadsheet application which also lowered the possibility of mistyping. A copied error, however, propagated errors to multiple cells. Participant B had trouble differentiating between using a copy operation or a relative copy/fill. This distinction appeared to be difficult for the student novices at first, but most students were able to understand both operations by the end of the course. Participant B often made the choice to enter each formula individually, as his use of copy/fill operations often did not have the results he expected. The small data set may not have motivated use of more generalized functions. He might have changed reasoning approaches with a much larger data set, where entering each formula individually became tedious.

The concept of absolute addressing was conceptually more difficult and participants had difficulties determining when and how to use this operation. One method of reducing confusion and making the spreadsheet application more readable was to use range names. Participant A always used range names, when possible, even from the beginning of the study. This use of range names appeared to be a concept she integrated into her basic knowledge and understanding. She also understood how range names could be used in place of absolute addressing. Participant C alternated between range names and absolute addressing. Based on the observations in the study, range

names did not fit into Participant B's localized view of the spreadsheet application, in which each cell represented a unique case of the problem.

Concerning spreadsheet skill and knowledge, there was general improvement in use of cell referencing, absolute referencing, and function usage. The participants could effectively use the descriptive statistical functions integrated into the spreadsheet. They seemed to view sum, average, count, minimum, and maximum as processes involving a spreadsheet tool or button and filling in a cell, rather than as a descriptive mathematical function. All of the students effectively referenced cells between worksheets by the end of the course. Most in the class were aware of more sophisticated functions for special processes such as amortization, but had not mastered this function yet. If the problems given differed significantly from those used in class, the students developed inaccurate representations of the problems. In the interviews following the think-aloud tasks, several of the participants thought that understanding of these more complicated functions improved with practice.

Research Question Three

Research questions three was: What learning experiences shape students' reasoning strategies that result in either correct or incorrect solution to problems? The students in this course experienced a variety of learning opportunities as content was presented in examples and lecture. In addition problem solving approaches were used that required various levels of direction and integration of chapter content, as well as prior course content. Individually, the teaching and learning methodologies were not effective.

If students focused on the step-by-step example-based learning in the chapters of the text, they produced an accurate spreadsheet as a product, but had little comprehension of what they did and were not able to apply the concepts to a different situation. Conversely, if the students initiated an ad-hoc solution process to problems given, the solution process was complicated with searching for appropriate commands, reworking formulas, backing up, and frustration. As a result, little learning resulted. The students were able to improve in spreadsheet skills with practice. Basic operation such as fill and copy were clumsy at first but by the end of the course, most students achieved some level of mastery. These operations reduced the number of errors in a spreadsheet by correctly translating an accurate formula to other series of data. In addition, use of fill/copy reduced the manual data and formula entry, thereby reducing the errors due to mistyping. If students did not learn how to use these operations effectively, errors resulted from improper use of the spreadsheet features. This result was also true of such features as *autosum*, absolute referencing, and the function insert tool. Participant A incorporated range names into her solution process when possible. This approach was effectively introduced early in the course that reduced errors and made the spreadsheet structure more readable to others. The other two participants and most other students in the class did not use range names as effectively as Participant A did. Another effective technique for reducing spreadsheet errors is to critically evaluate the results of a calculation. Participant B checked his results and it seemed had learned this technique prior to this course.

The participants' mathematical reasoning seemed to be in place prior to the course. This reasoning was a result of prior learning and experience. When the students

were not confident in their mathematics skills they tended toward trial-and-error approaches for developing formulas. They often took localized views of the single incident of a problem for one set of input values. Generalization was difficult for them as this complex process involved both mathematical reasoning and spreadsheet reasoning.

As part of the content of the course, students learned to do mathematical reasoning using the spreadsheet as a tool. If their mathematical reasoning was faulty due to prior learning, the faulty reasoning carried forward into the spreadsheet solutions. The fast feedback and flexible nature of the spreadsheet facilitated some of the painful arithmetic processes for students, so there may have been more incentive to try various solution strategies. However, the student might have simply tried many possibilities rather than reasoning, as evidenced in Participant B in Figure 4.2.

The results of the final think-aloud task give some insights into how learning during the class affected reasoning strategies. The students relied on means-end reasoning by learning to use the function insert tool for most spreadsheet functions. Use of the tool provides an intermediate goal in the solution of the problem. This approach also helps divide the problem into subgoals, as each expression for each parameter can be developed independently. As participant A noted, the tool also provides examples. As the function insert tool provides guidance the use of it might help reduce errors.

Appropriate and effective use of fill and copy operations helped reduce errors. Participant A used these operations effectively, while Participant B did not.

Learning to generalize results can help reduce errors in a couple ways. First, fill and copy operations can be used effectively, thereby transferring accurate formulas to

series of cells. Errors due to mistyping are reduced. Also, if the spreadsheet is implemented to accommodate a specific set of inputs, the spreadsheet may not be accurate if inputs are changed. This was the situation created by Participant B, as he developed formula for specific cases.

It should be noted however, that introduction of generalization involves introducing complex concepts in to the spreahseet solution. Many of these concepts are difficult for novices to understand fully and errors can result if the concepts are implemented incorrectly.

Another method that worked well to reduce errors was range naming. Participant A learned the technique early in the class. She only used it in cells that would require absolute referencing from another cell, but fill/down and copy operations were more successful as a result. Participant B did not use range naming and had difficulty with absolute referencing. He may have benefitted from trying to use range naming. Participant C did not complete the third activity, but in Task 2, used range naming to some extent but not as effectively as Participant A. It is notable that even Participant A did not use range names on the final exam.

The participants reported enjoying the ability in a spreadsheet to easily revise a formula if a result was incorrect or a modification was needed. They learned how to revise formulas and respond to the fast feedback of a spreadsheet environment. In several cases, participants evaluated results of a calculation for accuracy. They may have learned this in earlier courses as it was not stressed in this course. However, they often did not

check results for accuracy and part of a spreadsheet course should include developing this skill.

Chapter V

Discussion

In this chapter, the findings in Chapter IV are discussed and connections drawn between student reasoning errors in a spreadsheet course and the research literature on spreadsheet errors and reasoning associated with spreadsheet development. The research questions designed to examine these spreadsheet errors are:

1. What is student's reasoning that results in *domain knowledge* errors in spreadsheet solutions?
2. What is student's reasoning that results in *implementation* errors in spreadsheet solutions?
3. What learning experiences shape students' reasoning strategies that result in either correct or incorrect solution to problems?

A think-aloud protocol was used to study the reasoning used as participants developed spreadsheet applications. Retrospective interviews were used after each activity to probe for associations between reasoning and spreadsheet errors. Three student volunteers from an introductory spreadsheet concepts course completed three spreadsheet tasks outside of class during the five week duration of the course. The sessions were videotaped and analyzed to investigate the reasoning used during spreadsheet application development. Particular attention was focused on scenarios where errors were generated to determine the reasoning used that may have led to incorrect results.

One notable aspect of this research is the first examination of the reasoning category of errors in the taxonomy developed by Rajalingham, Chadwick, and Knight (2001) described in Chapters I and II. Powell, Baker, and Lawson (2008) noted that a recent literature review suggested that the reasoning category of errors has not been investigated and suggested studies such as the current one.

The focus of this study was an investigation of spreadsheet errors due to algorithmic, mathematical, or spreadsheet logic. Since the primary objective of the course in which the study took place was based on teaching spreadsheet skills and reasoning, it was assumed that the most learning took place in these spreadsheet skills and reasoning. While it did not appear that students gained new mathematical knowledge during the course, the course may have refreshed students' understanding of some mathematical concepts. Errors due to spreadsheet logic were most likely the result of faults in learning during the course while errors due to mathematical or algorithmic reasoning were due to learning prior to the course. Errors in spreadsheet logic may have been due to students' prior knowledge and skills, although an assessment of students' spreadsheet abilities at the beginning of the course indicated that they had little or no prior spreadsheet development experience.

This chapter begins with the general results of this study and their relationship to the findings of the relevant research described in Chapter II. The reasoning approaches used and errors generated while creating spreadsheet applications supported the findings of the research reported in the literature review. The research literature on errors associated with cognitive reasoning, word problems, learning computer applications, and

spreadsheets provided a framework for understanding and explaining the results of this study. The results of each research question are examined within the context of this research literature. The overall results of the current study are further discussed in association with the relevant prior research. Several ideas that emerged from the research process are then presented. Finally implications for future research and limitations of the current study are discussed.

Research Question One

The first research question considered students' reasoning when making spreadsheet *domain knowledge* errors. The participants' reasoning involved domain knowledge that included how they thought during algorithm development and mathematical implementation of those algorithms. The reasoning methods included trial-and-error, means-end, analogical, and case-based strategies. Only the first two were effective approaches for producing accurate applications, while the others by their nature lead to errors. At times, the participants also used strategies such as guessing, giving up, and asking for help indicating that they would probably not have created an accurate representation of the current situation on their own. This section discusses the results of the study in relationship to research literature on word problems, problem domain, cognitive processing, and generalization.

The real-world and mathematical reasoning used by participants in the study was the product of prior courses and experiences. Although the tasks used in the think-aloud

activities were chosen and developed to be reasonably domain-free, some concepts were better understood through real-life experiences and situational learning.

The tasks for the study were based on the spreadsheet course concepts and skills included to the point at which the tasks were administered during the five-week course. Panko and Halverson (2000) developed Task 1 as a domain-free problem for use in a number of studies. The assumption was that the participants would understand volume calculations and basic concepts of cost per unit. The spreadsheet concepts covered in Task 1 included data entry, formula creation, and formula fill/copy/drag. Spreadsheet Tasks 2 and 3 were increasingly more complex than Task 1 including material covered in the most recent class plus the basic material used in Task 1. The spreadsheet concepts added in Task 2 were absolute referencing, and the IF() function. The PMT() function and referencing between worksheets were added in Task 3. For most of the learning activities and tasks a starting spreadsheet template was provided so little formal spreadsheet design was involved. In most of the activities in the textbook chapter and end-of-chapter exercises, the correct answers were given. Students knew whether their results were accurate by comparing with the results in the text. In activities provided by the instructor, either the results varied depending upon students' choices for inputs or the correct answers were discussed after students had completed the spreadsheet. The participants had more practice with basic skills by the time Task 2 and 3 were administered. Although Task 2 was relatively domain-free, it assumed basic mathematical knowledge, conversion between units, and experience with gas mileage. The spreadsheet skills involved were data entry, fill/drag/copy, IF(), relative and absolute

referencing, and simple statistical functions. Task 3 was an income-expense application involving two worksheets with cell references linking them. The task assumed some domain knowledge of how overtime pay, payroll taxes, and benefits deductions are computed. The spreadsheet concepts that were added to those used in Task 2 were the PMT() function and referencing between worksheets. A section of the application dealt with computing a monthly mortgage payment and while many of the students were too young to have experience with mortgage payments, this material was covered in class. In addition, a homework assignment had provided additional exposure to problems such as this one. In many cases the algorithm and/or algebraic representations were given along with the instructions for a given cell.

All of the participants had taken college algebra and possibly some statistics; they indicated confidence in their mathematical ability when self-rating their background in arithmetic, algebra, and word problems to be fair to excellent (given a scale with *excellent, pretty good, fair, struggle with*, and *cannot do* as choices). While the mathematical processes associated with the course may have refreshed some knowledge from those experiences, it is doubtful that significant new fundamental mathematical learning took place during the spreadsheet course. New higher-level concepts may have been introduced, however, as the content of the course did include exposure to amortization computations as well as calculations involved in payroll computation. In essence then, mathematical errors in the spreadsheet tasks of this study were attributed to the difficulty in transfer between prior mathematics learning and the spreadsheet environment or to faulty learning earlier in the students' academic career. It did not

appear that any gaps in prior mathematical knowledge were bridged or faulty notions corrected as a result of the class or the research tasks, interviews and think-aloud sessions. The participants noted in the retrospective interviews that the mathematical reasoning was difficult for them.

One interesting shift in thinking regarded percent growth problems. When presented with a percent growth problem initially, the subjects used an algorithm for solution: base amount + base amount * percent increase. An alternative approach (base amount * (1 + percent increase)) was introduced in the text, used in hands-on activities in class, and presented in lecture along with the algebraic derivation from the initial algorithm. At least one of the participants began using this approach halfway through the class, indicating that the alternative algorithm seemed easier. This approach was considered a spreadsheet technique by the Participant A rather than the result of algebraic factoring.

Similarity to Word Problems

In many ways, the think-aloud activities were similar to solving word problem using a spreadsheet. The directions for Task 1 described a situation in which an estimate for the construction costs of a brick- or rock-wall needed to be estimated (Figure 5.1). Parameters for the problem were given in the one paragraph description. Participants needed to create the spreadsheet from scratch. In Tasks 2 and 3, a general description of the situations was given, along with a spreadsheet template to begin the problem solving

process. The template helped to maintain the focus on numeric content rather than layout and formatting.

Problem

You are to build a spreadsheet model to help you create a bid to build a wall. You will offer two options, rock or brick. The wall will be built by a crew of two, working three eight-hours days. The wall will be 20 ft long, 6 ft tall, and 2 feet thick. Wages are $10/hour/person. You will have to add 20% to wages to cover fringe benefits. Brick costs $2 per cubic foot. Rock costs $3 per cubic foot. Your bid must include a profit margin of 20% beyond your expected cost.

Figure 5.1 Problem Statement for Spreadsheet Task 1

In addition to the general description and starting template, a problem statement was included for each series in the spreadsheet. This problem statement was worded as a simple word problem (Figure 5.2).

Spreadsheets are a tool for solving word problems (Fiecht, 2002; Wilson, Ainley, and Bills, 2003; Stevens and Palocsay, 2003). In this study, the processes used in creating spreadsheet representations of the problems in each cell are similar to the processes described by Cook (2006) for college students and algebra story problems. This combination of human problem solver and computer-based tool can be effective, but the introduction of technology tools can also introduce detrimental effects. The spreadsheet acts as a cognitive tool or partner in the process, facilitating problem solvers to focus on the larger concepts rather than the arithmetic details and reducing the cognitive load due to performing the arithmetic operations. However, the visual and mechanical aspects of spreadsheet data maneuvering and manipulation can add to the cognitive load (Hendry &

Green, 1994). Brown and Gould (1987) reported that the cognitive processes involved in implementing a problem with a spreadsheet severely taxed working memory. In addition, the use of programming tools or computer applications has the potential for shifting attention away from the concepts within the problem to technical issues involving the tools. For instance, in the current study, participants often were more concerned with layout and colors of the spreadsheet than with the accuracy of the formulas.

Problem
The mileage spreadsheet contains mileage data for several makes and models of vehicles. You are to develop the rest of the application that computes several values based on fuel prices, fuel tank capacity, and mileage.

Instructions

Use **absolute reference/range names** and **fill/copy** operations wherever it is effective. If a **fill/copy** does not work the way you think it should, you may have to enter the formula for each cell separately.

1. Enter the formulas in the first row for the following
   - In cells F7, the average MPG in based on the min and max values in the two previous columns using the appropriate function
   - In cells G7 cost of driving a 200 mile trip based on average mpg and gas price in cell G2: Cost = 200/average mpg * Gas Price
   - In cell H7 calculate the amount of EPA discount. The Environmental Protection Agency offers a 5% discount in gas to vehicles with high mileage rates (this is hypothetical). **If** the average mpg is greater than 25 there is a 5% discount on the Cost of going 200 miles. Create a formula to calculate the amount of the discount. You should use an **IF** function.
   - In cell I7 enter the formula to calculate adjusted cost for the EPA discount: Cost of driving 200 miles – EPA discount.
   - In cells J7 enter a formula to calculate the distance that can be driven on a tank of gas, based on the average mpg and tank size.

Figure 5.2 Excerpt from instructions for Task 2

The initial steps in translating a problem into a spreadsheet representation are similar to those for solving word problems. In spreadsheet application development, a problem statement must be translated into an intermediate mental mathematical representation prior to being implemented into a spreadsheet representation. Kintsch (1998) identified two steps in solving word problems that gave students difficulty: formulation of a mathematical model of the problem and determination whether or not a correct model had been formulated. Koedinger and Nathan (2004) described two phases involved in solving word problems: a comprehension phase and a solution phase. In the comprehension phase, the text of the problem was translated into an internal representation of the quantitative and situation-based relationship. During the solution phase, internal and external representations were transformed into a solution. In spreadsheet application development, a problem often needs to be translated into an intermediate mental mathematical representation prior to being implemented into a spreadsheet representation. The solution phase may thus be different in a spreadsheet environment than in a paper and pencil algebraic environment. Some researchers (Wilson, Ainley, and Bills, 2003) have found that the spreadsheet environment allows for faster feedback and flexibility in developing solutions. Cook (2006) found that with word problems, even after successful intermediate representation college students formed incomplete or inaccurate mathematical solutions to the problem proposed in text. This result was identified in the spreadsheet representations of the current study. The participants often seemed happy to have a numeric result, even though part of the formula was missing. In the current study, the participants indicated that the spreadsheet

environment made solving these problems easier or more fun. Part of the reason for their statements was that if they made an error, or needed to adjust part of the model, it was easy to do by simply changing specific cell values for the spreadsheet to re-compute the solution.

When reasoning through the problems, the subjects read the problem statement for a particular cell and then scanned the spreadsheet for the factors and terms needed in the formula. Jonassen (1995) noted that spreadsheet developers must identify patterns and relationships among the data they wanted to represent. Next, the relationship was modeled mathematically, using rules to describe the relationships in the model. In the current study, after scanning the spreadsheet for components of the formula for the current problem, the subjects entered a formula confidently, initiated a myriad of reasoning approaches, or quit. In the first case, the formula was often provided so little mathematical reasoning was needed.

If the participants were unable to develop a solution intuitively, they began a reasoning process towards a solution. They talked in terms of the problem while scanning the spreadsheet, often repeating to themselves and moving the cursor over a prospective factor.

If a part of the problem matched something they recalled from class, they tried to integrate that solution into the current problem. They often re-read the problem or last statement, as students often do in solving word problems (Cook, 2006). Rather than subdivide the problem into manageable parts, they often chose pieces of the problem that they could associate with labels and cell data of the spreadsheet. Cook observed college

students *number grabbing,* selecting all numbers in a problem statement regardless of relevance, while solving word problems. Cook also reported that students solving word problems comprehended the problem better if they could remove extraneous information from the problem description. In the current study, while the problem descriptions for each cell contained only relevant information, the work area (spreadsheet) contained all previous factors as well as some in the template that had not been used yet.

During the think-aloud processes in the current study, mathematical reasoning included murmuring about factors, consideration of multiplication versus division, and deciding between subtraction or addition. Many of these bits were implemented into a portion of a spreadsheet formula in an iterative fashion. Formulas were adjusted quickly and flexibly as the participant worked through the problem for a cell. They often paused and questioned whether the solution thus far was right or not, often hoping for comments or help from the researcher/instructor rather than making a conceptual evaluation of the result value. Reasoning often began as means-end and deteriorated into trial-and-error, guessing, and quitting. The participants in this study often resorted to line-by-line translation of a problem, retracing their steps, and re-reading instructions, much as in solving a word problem (Cook, 2006). Stevens and Palocsay (2003) reported similar behavior with students solving word problems using a spreadsheet. This type of approach led to errors if any part of the problem was not translated effectively into an appropriate mathematical representation.

The mathematical reasoning in the spreadsheet problems was difficult for the participants and often appeared to fail in following a logical process, particularly when

the algorithm or formula was not provided. In many cases, where an error occurred, the spreadsheet formula was a valid formula and generated a numerical result, but the formula did not accurately model the problem, as reported by Panko and Sprague (1998). In the current study, the participants were often happy with any numeric result. During Task 2, errors resulted from the participants continuing to use a factor that was part of an earlier cell computation, but not required for the current cell. Anderson (2000) noted similar behaviors in solving algebra problems and were attributed to faulty analogical reasoning. There were also problems with basic algebraic skills such as deciding between operators and parentheses for grouping.

In some cases, the participants assumed a problem was insolvable and quit. Hendry and Green (1994) reported that even experienced spreadsheet users sometimes concluded a problem was insolvable when they saw no way to approach it. The subjects in their study behaved as if they were novices and began with trial-and-error type approaches. In the current study, the students may not have believed the problem was impossible to solve, but felt they could not solve it, given their current knowledge. The overtime pay problem in Task 3 was an example of such a case. This problem appeared at first glance to be fairly simple, and the solution was not very difficult for a given instance. Generalizing the solution, however, was difficult and required reasoning that was different from conventional algebraic thinking.

The participants usually did not check to see whether or not their answer was reasonable. Kintsch (1998) observed with word problems that, even if a participant did check to see if the result was reasonable, it was difficult for them to decide whether a

solution was accurate. In the current study, Participant B inspected results of calculations more often than the other two participants, but it was not clear whether or not he had an idea of what reasonable was. He seemed to have more of an intuitive feeling for how big a rock wall would be and for gas mileage concepts. This student was older than the other two students and had more work experience/life experience. However, his results were wrong as often, if not more, than those of participants A's and C's. Koedinger and Nathan (2004) proposed that situational learning (real-world knowledge) contributed to accuracy, but that prior learning must be able to be transferred to the new situation.

Problem Domain

One of the areas investigated in this study was spreadsheet errors associated with the domain of the problem. In the current study, the expectation could have been that the domain errors were minimal, as the tasks chosen were similar to context areas familiar to the student spreadsheet developers that did not require a great deal of area-specific knowledge to understand. However, it was found that the students' domain knowledge was more limited than expected. Hendry and Green (1994) found that if the spreadsheet formulas closely matched the requirements of the problem domain, creating that cell formula was easy. In this study, there was a disconnect between previous mathematical knowledge and using that knowledge to solve real-world problems This result could have been due to an inability to transfer knowledge learned in mathematics courses to a new environment. Another possibility was that the students never gained a deep understanding

of the concepts involved in the first place and had learned only enough of what was needed to fulfill the requirements of the previous courses. When asked what was hard about developing a spreadsheet application, both participants A and C indicated the mathematical part was difficult. Participant A did note that she probably did know more math than was evident in the study, but did not have the confidence to use it.

Some problems in the think-aloud activities appeared simple at first, but were not solved easily, requiring combinations of functions. The solution may have required specialized spreadsheet functions (such as PMT) that might have been missing in the user's knowledge for solving the problem within the domain of work. Hendry and Green (1994) noticed this same phenomenon with experienced developers. After working with a few examples, however, the students in the class began to develop concepts of what appropriate values were for a monthly mortgage payment so they could then assess the accuracy of their formula.

Omission Errors and Cognitive Overload

A common spreadsheet error is that of omission, leaving out an important item in the problem solution. A part of the problem is omitted from the spreadsheet representation of the model. In Panko's (1996) study with the "Wall" problem (Task 1 in this study), participants omitted the fact that the problem described two workers building the wall and so accounted for only one worker. However, the students in the current study often left terms out of a formula rather than out of the entire model. One error that was common was in conversion between units, for example, converting an annual figure

to a monthly one. Panko (2000) observed similar errors in a study of students developing spreadsheets for two relatively domain-free problems and suggested that a source of these omission errors was cognitive overload. The researchers proposed that if the amount of information was larger than short-term memory, a "blend" error might result, containing elements of competing statements. This phenomenon might explain the reasoning used by participants in Task 2. In several cases, the participants used factors from the model that did not pertain to the current cell. They also used only two factors in a formula that required three to be correct. This omission could be due to a hesitation to try a more complicated formula or perhaps the cognitive overload limited their reasoning.

Generalization

One of the benefits of a spreadsheet is the ability to represent data as variables and to generalize the application to accommodate a spectrum of data and conditions. Inputs can be changed and the spreadsheet produces correct results for the new set of data. Creation of a formula to accommodate a variety of data and possible conditions is more complex than developing a formula for a single case and thus is more difficult for the students. In the current study, the participants sometimes developed individual formulas for each cell in a series, rather than developing a generalized algorithm and formula for the series. Values were sometimes hard-coded rather than relying on cell references. While the results may have been accurate for that set of values, two possible problems arose from this approach. The first problem was that the increase in typing potentially resulted in more mechanical errors. The second problem was that even

though the data may have been correct for this set of numbers, if any of the series inputs changed, all formulas likely required adjustment.

Two possible explanations existed for the use of a non-generalized solution such as the one described above. First, the participant may have held only a local view of the model. During the spreadsheet course, while the students developed many spreadsheet applications, they were typically only asked to use them once and were not asked to adapt any of the problems for other cases. In addition , the students may have felt that it was easier to develop separate formulas for each cell in the small models used in the course and the research study tasks rather than learning to generalize the formulas. Dealing with a large number of cells for a series may be necessary to motivating developers to master the skills necessary for more sophisticated referencing techniques rather than creating each formula individually. Feicht (2000) found that seventh grade students tried to generalize when the number of cells in a series was over ten.

Research Question Two

The second research question was focused on identifying the students' reasoning when making spreadsheet implementation errors. Errors made during spreadsheet implementation included results of faulty copy and fill operations, referencing operations, and improper use of spreadsheet functions. This result is similar to that of Brown and Gould (1987), who found that most of the errors created by experienced spreadsheet developers were formula errors, including references to the wrong cells, cell reference errors caused by copying, and confusion over formula and values. Most of the errors that

result in bottom-line numerical errors are logical errors in the formula. The spreadsheet implementation errors also include syntax errors, but the spreadsheet software usually catches these errors. This section discusses spreadsheet logic errors organized into sections based on the spreadsheet operations involved: copy/fill, referencing, aggregate functions/simple statistics, and spreadsheet functions.

Copy/Fill

Copy and fill operations were some of the early topics covered in the spreadsheet course for this study. Initially, the participants had problems in distinguishing between fill and copy commands. The participants occasionally copied the contents of a cell rather than using a fill operation where the formula is copied and adjusted for the selected cells. This difference in the operations was subtle at first, involving pointing the cursor at the middle of the cell (copy value) or the lower right-hand corner (copy and adjust formula). The errors resulted in all selected cells having the same value and usually the participants caught this error. The errors resulting from using relative addressing where absolute referencing was needed were also usually identified, as the solution results were far beyond reasonable values (negative values where only positive should be considered) or dividing by zero errors. The participants often had trouble knowing when and how to use absolute referencing versus relative addressing. They often erred on the side of absolute referencing and constrained all the variables or the wrong ones. Hendry and Green (1994) found that a common source of error was using the incorrect referencing technique. One experienced user in his/her study recalled having problems with copying

functions early in the course, but improved with more experience. A similar improvement in basic skills was observed for the participants during this study. By the end of the course and Task 3, all three participants were competent with the fill and copy operations.

Referencing

Two of the participants created fewer referencing errors and used more absolute referencing in appropriate places than the third participant. One participant in the current study used range names effectively, thus reducing errors due to absolute referencing. Janvrin and Morrsion (2002) had reported that spreadsheet accuracy increased due to use of range naming for all referencing. Another participant did not use absolute referencing. He made an initial attempt unsuccessfully, but then resorted to entering individual formulas for all cells.

Aggregate Functions and Simple Statistics

A few errors during the think-aloud activities were identified with aggregate and simple statistical functions such as sum, average, min and max. These functions are similar in that they all involved performing an operation on a range of cells. For these functions, the participants usually used the *autosum* function or the function-insert tool. The *autosum* tool potentially generated errors based on the default range selected. One participant did use *autosum*-average incorrectly by keeping the default range selected by the function. This error then propagated through the rest of the development of the model. When the participants needed to use a function more complex than basic

aggregate or statistical functions, they almost always used the function insert tool rather than developing the formula manually. The function insert tool may have reduced errors by providing a template for the proper number of arguments with examples. The review of the research literature indicates that the impact of the use of *autosum* or the insert function tool on spreadsheet errors has not been investigated.

Although participants became adept at aggregate functions and simple statistical functions, they had difficulties with more complex functions that required multiple parameters. This gap in ability may have been the difference between developing skills and developing a conceptual understanding of the spreadsheet representation of the data elements of the problem.

Spreadsheet Functions

The functions that were more complex than simple statistics or aggregation functions were IF(), for decision making, and PMT(), for loan amortization calculation. Both were difficult for the participants and numerous errors were generated in association with these functions. Some of the errors resulted from faulty mathematical reasoning within a parameter for the function such as inequality operators and percent concepts. One problem with the IF() function was the logical reasoning about how to enter the proper form. If the wording of the problem matched the logical order of the IF() statement, the implementation went smoothly, although mathematical operations on parameters were difficult for the participants and errors resulted. One problem the participants and class experienced in general was that the IF() function returned a value

even if only one or two of the three parameters were entered. The results may have been accurate for certain instances of the problem. In the retrospective interview with Participant A in Task 2, she indicated that she felt her abilities with the IF() function would improve with practice.

The IF() function can be used to generalize a function when criteria are present. In the study, occasions were evident when students did not use an IF() to create a generalized version of the algorithm; rather they dealt with each case individually. Also, they incorporated hard-coding within the IF() parameters. As with other qualitative errors, these errors may not have resulted in a bottom-line error, but might if the input values were changed. An example is shown in Figure 5.3 with a portion of Participant B's solution for the payroll worksheet in Task 3. The Gross Pay and State Tax computations should use IF() statements to meet all possible conditions, but this student developed separate formulas for each. The significance for this study is that the student's lack of understanding of the capabilities of the spreadsheet resulted in methods of implementation that potentially lead to errors.

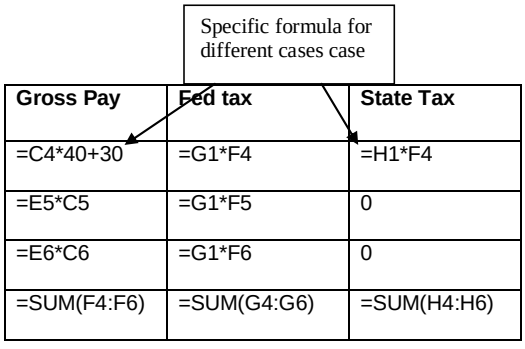


Figure 5.3 Portion of participant B's non-generalized solution for Task 3

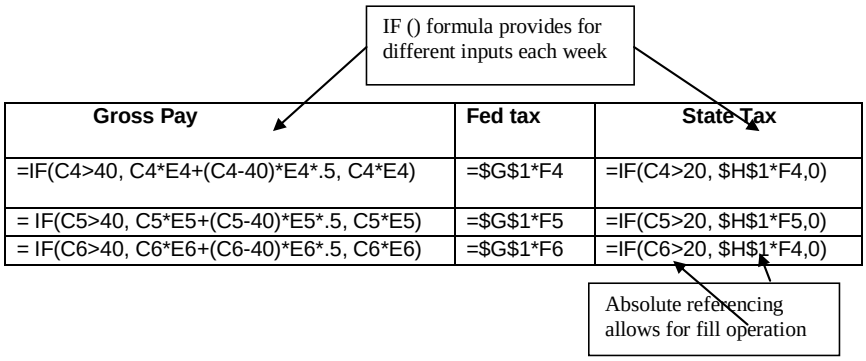


Figure 5.4 Generalized Solution to portion of Task 3

The most complex formula, PMT() for amortization of loans, was introduced in the last two weeks of the five-week course. This function was considerably more complex than the others encountered in the course. The real-life knowledge, algorithmic, and mathematical knowledge involved were more abstract than previous topics. As with the IF() function, PMT() required three arguments (rate, terms, amount). But the arguments were not in a familiar logical sequence (such as If, then else). The function tool helped with associating function parameters with spreadsheet cells, but the student needed to know when and how to use the function. Most of the errors with this function were due to improper scaling, arguably a domain-knowledge error. Two students, including PParticipant B, did not use the PMT() function on the final exam, but developed a reasonable, although inaccurate formula. The participants did not seem as confident that practice with this function improved their accuracy. They assumed that if they had to use this function in a business-related spreadsheet application, expert advice would be available to provide guidance.

Research Question Three

This section discusses how learning experiences shape students' reasoning strategies regarding spreadsheet errors. The instructional and learning process for the course used in the study is summarized, followed by sub-sections for domain errors, mathematical errors, spreadsheet errors, error-making and learning, and the participants' perspectives of learning with spreadsheets.

When students learned spreadsheet skills and concepts in this five-week college course, they were integrating new information into existing knowledge, the result of previous courses, experience, and perceptions. Since many of the concepts and features of spreadsheets involved modeling real-life situations mathematically, part of the content of the course included how to perform mathematical computations using a spreadsheet as well as mathematical reasoning using spreadsheets. For instance, the students learned the spreadsheet skills and concepts to achieve the goal of averaging a data in cells in several different ways, depending on the situation (i.e. add the values in n cells and divide by n, use the AVERAGE() function, *autosum*, or the function insert tool). New spreadsheet concepts were added each week of the course, and the associated activities built on and included previous material covered to help reinforce that learning. Most basic spreadsheet concepts were repeated in a variety of situations to guide students' developing understanding.

All students had prior courses in college algebra and a high school mathematics background. Although the objectives of the course did not include teaching mathematics, there is the possibility that students refreshed some prior mathematical learning

experiences through of them use in the course. They may also have learned mathematical concepts that had not become part of their knowledge base in prior mathematical experiences as a result of using spreadsheets to facilitate mathematical reasoning. Efforts have been made to integrate spreadsheets into mathematics courses and teach mathematical reasoning using spreadsheets in recognition of proposed benefits of using the spreadsheet environment (Wilson et al., 2003; Jonassen, 1995). The students' understanding of algorithms and mathematical content used to describe content domains might have been enhanced through the identification of values and developing formulas to interrelate values in a spreadsheet. Jonassen (1995) described this phenomenon suggesting that the use of a spreadsheet made the underlying logic obvious to learners and improved their understanding of the interrelationships and procedures. Also, in the spreadsheet environment, the underlying logic supported a graphical representation. While these features may have helped in learning, most of the students' mathematical knowledge was assumed to be based on earlier experiences and any mathematical misconceptions or knowledge gaps that contributed to errors. In the course and during the think-aloud sessions, mathematical reasoning was a difficult process for the participants in the study and led to many errors.

In the course this study was based upon, initial spreadsheet concept instruction was through step-by-step examples from the textbook and through instructor lectures. A problem-solving approach was also implemented through exercises from the ends of the chapters designed to apply the new spreadsheet skills. In addition, activities developed by the instructor were used to promote meaningful learning, described by Mayer (2002)

as learning leading to transfer with a resulting effect on new learning. Mayer also noted that learning might not take place if learning amounts to acquiring a collection of specific responses to situations. Students may have viewed each spreadsheet problem or used the spreadsheet features individually rather than thave tried to develop new knowledge about how and when to use the spreadsheet. This phenomenon might also have accounted for problems with mathematics stemming from previous learning. The students may have learned mathematics in a way that enabled them to solve specific algebra problems, but without sufficient understanding to apply to new situations.

The students in the course processed the step-by-step instructions for examples in the chapters almost mechanically. The problems at the end of the chapter and those provided by the instructor had varying degrees of directions and explicit instructions. Students encountered increasing difficulties and made more errors with those problems that had little or no explicit direction and that had no correct answers provided. During the think-aloud activities, the participants had trouble translating Task 1 into a spreadsheet solution as no template was provided. In Task 2 and Task3, starting spreadsheet templates were provided but the participants had difficulties translating the instructions for individual cells into spreadsheet formulas. Reimann and Schult (1996) reported that novices had trouble deciding which next step to take in the translation process. They noted that even when an example was explicitly provided, mapping to the correct representation and modifying the solution were difficult. Students often mapped syntactically from the example to the result, leading to ineffective and possibly erroneous solution attempts. In the class activities, the final exam, and Task 3, errors

resulted when students tried to match the parameters of the PMT () function with previous examples of the PMT() function.

Gick (1986) referred to learning strategies such as learning-by-example and problem solving as indirect methods of accelerating schema learning. The learner had to induce strategic knowledge from previous examples. They may have developed knowledge and strategies that lead to accurate solutions for particular cases. However, the same reasoning may have resulted in errors for some situations. Moreover, spreadsheet results can be obtained in the absence of learning. In the current study, this phenomenon was evident in student learning of the absolute referencing techniques, the IF() function and PMT() function. The students had trouble learning when and how to use absolute referencing. In the case of absolute referencing, the course curriculum provided a variety of situations where the students encountered situations where absolute referencing was effective. The objective was to give enough exposure and experience that the students understood the concepts behind referencing techniques to facilitate meaningful learning. However, it seemed that in many cases, the students might have done what was needed in that specific case to get a result. Analogical transfer may not have taken place because students failed to recognize structural similarities (Reimann & Schult, 1996).

The participants in the study and students in the course all had difficulty with the IF() and PMT() spreadsheet functions, resulting in many errors. Similar to the educational design used to teach absolute referencing, in the situations designed to help student learn the IF() function, a variety of situations were used to assist the learners in

extracting central concepts. The problems used included situations with simple logical reasoning and corresponding parameters, complex reasoning and multiple term parameters, and numeric and textual data. In all cases, however the underlying logical reasoning was based on the sequence *if* <*condition*> *then* <*action*> *else* <*action*> that most students had familiarity or comprehension. In contrast, the situations in cells requiring a PMT() function were similar, but the underlying concepts were abstract and difficult for most students to understand. The participants in the study felt they could improve on the IF() function with more practice and appeared to be using logical reasoning when implementing the function. They did not seem to feel as confident with the PMT() function and when solving a problem requiring PMT(), the approach used was trying to choose appropriate parts of problems from previous examples in class.

Learning and Domain Knowledge Errors

Little new domain knowledge was introduced in the course. Payroll and amortization concepts were the only information presented that was beyond basic mathematical concepts. Payroll concepts were used as the students were somewhat familiar with those concepts; these concepts provided several opportunities for integration of spreadsheet concepts including the IF() function. Amortization was introduced to give students an exposure to a more complicated spreadsheet function, PMT().

The tasks used in the study were relatively "domain-free" in that no specialized real-world experience was expected other than what had been covered in the course. The

first task, the "Wall" problem has been used in several other studies and was developed by Panko and Halverson (2000) as a domain-free activity. The second and third tasks were similar to problems used in other studies and tailored to the content of this course. The second task assumed a basic ability to convert between units and a real-world feeling for data involved in fuel economy. The third task was similar to the final exam for the course which occurred prior to the think-aloud sessions.

Several of the real-world errors involved the concepts introduced in the course, payroll and amortization. However, many of the actual errors involved mathematical reasoning. If a participant did not include an adjustment factor from annual percentage rate on the interest of a loan to monthly, was the cause rather than not remembering to convert, not understanding APR, or simply having too much going on in working memory.

Despite efforts to minimize domain-content, several errors involved real-world knowledge that occurred during the think-aloud activities. Participant C had trouble recalling the algorithm needed to compute volume in the first task. All participants had difficulty developing algorithms for gas-mileage computations in Task 2 with errors resulting initially. The participants also had difficulty developing a generalized formula to calculate overtime pay. Most of their initial attempts compute total pay resulted in formulas that may have been accurate for specific cases, but were not able to be generalized.

Based on the definitions of categories in the taxonomy used for this study, an argument could be made that an error made during the practice task was a real-world

knowledge error. Here, all participants used the wrong algorithm to calculate the value of a set of coins. They immediately saw that the answer was wrong when questioned, but had difficulty determining how to develop an algorithm. A similar problem was part of the final exam and there were no errors. Either the students had learned to be more discriminating in the use of spreadsheet functions or the same visual cues that were in the practice think-aloud activity were not in the final exam.

The errors resulting from real world knowledge may have been reduced if the participants were more familiar with the domains of the problems. In a retrospective interview, Participant A indicated that if she had to use a spreadsheet in a job, they would check out the formulas with someone prior to development and then check the results against test data.

Learning and Mathematical Reasoning Errors

The students learned how to do mathematical computations using a spreadsheet. The spreadsheet removed the mental burden of arithmetic tasks (Ozgun-Koca, 2000) and may have allowed a student to focus more on the mathematics. However, the spreadsheet produced results, even when the formula entered did not reflect the problem at hand. The participants seemed to enjoy doing mathematics with the spreadsheet, but had great difficulty establishing mathematical representation of the problems. The mathematical reasoning often began with means-end and deteriorated to trial-and-error. Theparticipants tried various combinations of variables, were confused over percent

concepts, guessed at operators, and were reluctant to move beyond simple two variable solutions.

Participant A had difficulty determining the correct placement of parentheses for grouping and admitted this action was always a problem for her. The participants all remarked that the mathematical part was difficult for them. Yet, all had taken and passed several math courses, including college algebra. Seemingly, either the knowledge they had accumulated in previous math courses did not transfer to a new environment, or they never developed a deep understanding of the mathematical concepts in previous courses.

When asked what was difficult about learning to solve problems using spreadsheets or developing spreadsheet applications, the participants reported that the mathematical part was difficult. They typically used statements such as "I've never been good at math" and "The hard part is putting it into equations." It appeared that while they were able learn to do some mathematical reasoning using the spreadsheet, either their prior knowledge was not adequate to translate a problem into a mathematical representation or they did not learn how to transfer their previous mathematical knowledge to a spreadsheet environment. Any further efforts to investigate this issue should include a pencil-and-paper mathematics assessment solving similar problems prior to the study. The results of the assessment would give an indication of whether the developer could solve the problems without a spreadsheet.

Learning and Spreadsheet Errors

Part of learning new any new software package requires learning how to maneuver around the various menus and user space. At the same time, with a highly interactive package such as a spreadsheet, the learner is developing skills and conceptual knowledge regarding representation of some situation in this environment. A new developer does not know how to use the components of a spreadsheet, manipulate data, enter formulas, etc. As they begin to understand these operations, some become skills which they can perform without thinking about them.

Both a skill component and a conceptual component to the new knowledge are involved with spreadsheets, as with any other computer application. Hand and eye coordination are for visual cell manipulation as well as establishing relationships for formulas. As the students began to use several new skills, their movements and operations were sometimes clumsy and generated errors. In particular the basic skills such as cell entry, using an equals sign prior to a formula, drag/copy techniques improved quickly in the first few weeks of the course. Implementation of the function insert tool and *autosum* evolved from new learning accompanied with unsuccessful attempts with revisions to reliable skills. Riemann and Neubert (2000) described this phenomenon in reference to computer users learning new skills. However, the fact that some of these operations may become automatic is a concern. Drag and copy operations perform complex iteration over cells. In a traditional programming language, one needs to write a looping procedure and maintain awareness of the details of the processing taking place.

In the spreadsheet environment, copy and fill operations have attributes similar to those in painting or graphics software packages. The participants did not appear concerned with the internal formulas during the copy/fill. They did reflect on the result and revised the formula. Their initial attempts at copy and fill required effort to hit the mouse on the correct spot. Clumsy mouse movements resulted in formula copy operation going awry. Practice improved the copy process, but even when this operation was performed well, little forethought was given to how the formula translated. The exception was when the variables have been implemented using range names.

The operations of sum, average, maximum, minimum, seemed to become push-button actions using the *autosum* icon, ∑. This spreadsheet feature required practice. Initially, the participants often clicked on the ∑ icon and then chose which aggregate function to use. At this point, the aggregate functions chose a default range and the user accepted by releasing the mouse. The students in the class and participants in the study had difficulty initially realizing the "responsibility" associated with these mouse clicks. They also had trouble learning to correctly designate the target or destination cell. It did not appear that mathematic reasoning surrounding the concept of average was involved. In this case, the error may be difficult to categorize as either the developer used the wrong algorithm, or they used the correct algorithm but included extra values, or did not understand how the autosum capability of the spreadsheet functioned. It seemed that this type of error is a spreadsheet logic error, since as mentioned previously, the students were not thinking of the mathematical representation. The same error may not have been made if the student used an alternative method for computing average that more closely

resembled the mathematical concept involved. This might be an area requiring further study.

Spreadsheet functions were new concepts to the students in the course. While some were familiar concepts (i.e. SUM() , AVERAGE()), functions such as IF() and PMT() were not. The initial action used frequently by the participants with most functions was the function insert tool. This method was introduced in both the textbook and class activities for using functions. The approach provided a means-end reasoning strategy that helped the student developers get closer to their goal. Once in the function insert tool, they were prompted for parameter and examples were given.

When learning the new spreadsheet concepts, the participants did not effectively use the new information until they gained some experience. Initial attempts resulted in failures and revisions and students became proficient with the "undo" feature. Riemann and Neubert (2000) referred to this action as "weak hill climbing." In their description of people learning new interactive computer applications, weak hill climbing was eventually replaced by knowledge-based strategies. The environment still mainly triggered subgoals, but terminal actions were known and did have to be discovered. Riemann and Neubert (2000) also noted the difficulty of learning in a complex graphical environment, since many changes occurred simultaneously, some relevant for the main event of the action and some not. The students in the course were used to complex graphical environments, but it was difficult for them to find everything they wanted without practice and to keep track of all that was going on as a result of a data manipulation and use of functions. A specific example was the use of *autosum*. The participants had

difficulty learning how to effectively set the target cell and range of values to act on. Initially, the wrong range was often selected or the target cell ended up being not what was needed and error resulted. Also, the spreadsheets used in the class and in this study were small enough to generally fit on one screen, whereas most real-life spreadsheets are much larger than the screen and involve multiple worksheets. In the later activities in the course and in Task 3 of the think-aloud activities, students did have difficulties managing references between multiple worksheets because they could not see everything happening and the scope of their actions was beyond the localized cell environment where they worked.

Error-making and learning

Making errors can be part of the learning process. Participant A remarked that she learned from her errors. If the learners are monitoring their progress and assimilating the error and its revision into their knowledge base, their future development would be more accurate. Kolonder (2001) noted that errors may be considered necessary in case-based learning since subjects needed to attempt to apply what they thought was applicable; their failures focused attention on subtleties that they had not previously recognized.

Anderson (1989) discussed the role of errors in analogical reasoning in arithmetic and algebra problems. Errors were classified as slips, as importations of prior misconceptions into a new domain, and as within-domain misconceptions. A slip was characterized by not being consistent where the subject was able to self-correct when the error was identified. This case happened in many of the simple errors students made, for

instance the scaling errors. In Andersons' study, slips appeared to increase in frequency with working memory load and decrease with practice, which added support for the idea that these local omissions may have been due to cognitive overload.

Anderson (1989) did not deal with errors in the second category because the subjects worked primarily in one domain with few prior conceptions. However, in the current study, reasoning involved mathematics and real-world knowledge beyond the spreadsheet domain. There did not seem to be a consistent misconception that was integrated into the participants' spreadsheet applications. That is, the mathematics errors they made were inconsistent and real-world knowledge was lacking rather than in error. An argument may be made since the major function of the spreadsheet is representing mathematical relationships, that mathematics is not a separate domain from the spreadsheet domain. This idea appeared to be the case from an error-based perspective when considering misconceptions within-domain.

Andersons's (2002) study focused on misconceptions within-domain as a consequence of the learning that took place in the domain since he felt these errors reflected the learning process. The errors included those that resulted from students' attempts to bridge points in their problem solving where they had inadequate knowledge. A permanent misconception resulted if subjects believed their repairs. If subjects invented the repairs for a particular instance, an inconsistent pattern of errors resulted. In the current study, the problem was that the participants made inconsistent errors when using hard-coding of variables and absolute referencing errors.

In Task 2, which required use of several data elements of the spreadsheets in several different formulas within the spreadsheet, participants used data and solution techniques from one spreadsheet series in a series adjacent to it. Anderson (1989) had noted similar errors and felt they were due to flaws in analogical reasoning.

Chadwick and Sue (2001) suggested that instruction should include how not to make errors. As part of learning, new information is integrated into the knowledge base as a schema, a mental data structure (Gick, 1986). Reason (1990) noted that a schema contained evidence of how a particular recollection should appear, not a representation of what it should not look like. Systematic errors can result from fitting the data into the wrong schema, from employing the correct schema but filling gaps with best guesses, and from relying too heavily on active or salient schemata. As noted in Chapter II, a survey of current spreadsheet textbooks indicates no content regarding spreadsheet errors, other than an introduction to Excel's auditing tools.

Perspectives in learning to use spreadsheets

The participants reported that the spreadsheet was a positive method of solving problems and mathematical reasoning due to the quick feedback and ease of revising formulas. Students enjoyed the flexibility of the spreadsheet and the ability to make a mistake and revise. Wilson et al. (2003) found that the quick feedback of a spreadsheet encouraged students to adjust their formula to achieve success. While this phenomenon was observed in the current study, another possible negative effect were suggested. If the participants did not critically reflect on the cause of the inaccuracy of the result, the

feedback may have just prompted them to try something else. Mayer (2002) noted that during meaningful learning, the learner was actively engaged in the process, formed hypotheses, tested the hypotheses, kept correct hypotheses, and selected new ones for incorrect answers. Learning involved making sense of the learning situation and feedback helped build rules or procedures. The feedback itself did not promote learning, but the learners' interpretation and understanding of the feedback. Ainley et al. (2003) added that judging whether feedback was reasonable and considering what how the spreadsheet responded in its calculation was important. The participants in the current study often used a trial-and-error approach, revising a formula several times. In particular, during mathematical reasoning, it did not appear that whatever errors they made, that the knowledge was being integrated through the learning process. Feedback from the spreadsheet and errors associated with basic spreadsheet skills did seem to help improve the accuracy of spreadsheet. In the more complex operations such as absolute referencing and three-parameter functions, the feedback contributed more to a trial-and-error approach, with errors contributing little to new learning.

In an interview after a think-aloud session, Participant A stated that she learned from making mistakes and after seeing her errors, she did not usually make the mistake again. While it was encouraging that she wanted to learn from her errors, there were some consistent errors in the spreadsheet applications she developed.

While participants reported enjoyed working with the spreadsheet and enjoyed learning new skills. Many of the operations were like solving a puzzle and "painting" in that, the goal was to fill in cells and dragging one cell to other ranges. Lithner (2000)

reported that students solving calculus word problems treated problems as puzzles that required the selection and application of some operations. Similar to observations in the current study, Lither noted that students used a combination of matching similarities between what was familiar and the current problem and mimicked solution procedures rather than understanding mathematical concepts. In his study, the students employed strategies and solutions that were familiar based on surface features rather than on even elementary mathematical reasoning and accuracy. If a solution fell outside a prescribed solution process, errors occurred in local solution steps. In the current study, memorizing a development process for any single spreadsheet application was difficult for the purpose of basing a later development on that memorized information.

Overall Results

The participants of this study made domain-knowledge errors as well as spreadsheet logic errors. The quantity and types of errors were consistent with previous research (Brown & Gould, 1987; Panko & Sprague, 1999). Each spreadsheet developed as part of the study contained errors. The participants' errors seemed representative of the class within the instructor's experience with college students learning to develop spreadsheet applications. In the class activities, most spreadsheet development that was not directed by explicit step-by-step instructions contained some errors. In fact, on the final exam for the class, all spreadsheets (n=13) had at least one error leading to a bottom line spreadsheet inaccuracy. Panko (2005) reported that the percent of all spreadsheets having errors was possibly as high as 90%.

The errors that participants made were diverse, consistent with Panko and Sprague (1999) study with students when developing the same domain-free spreadsheet application used in Task 1. Those researchers had expected to find several common errors, providing target areas for error prevention. Instead, the results indicated a variety of logical errors among subjects. They suggested that students may not make the same error in different spreadsheets. Since the current study involved multiple spreadsheets, this phenomenon was observed, however the students had additional spreadsheet instruction between each spreadsheet development session.

Categorizing the errors in the current study using the taxonomy developed by Rajalingham et al (2003) was difficult. This research was the first effort to apply the reasoning categories of the taxonomy to spreadsheets under development. The categorization for many errors was straightforward. If a student did not think of a correct algorithm or used an incorrect algorithm, the error was classified as a real-world error. Mathematical errors were those that occurred after the correct algorithm was chosen, but implemented incorrectly. Participant A was an example when using parenthesis incorrectly for grouping of terms in Task 2. Spreadsheet errors were attributed to improper use of a spreadsheet function or operation. These errors included copy/fill operations, referencing errors, difficulties with IF() and PMT(). Several cases, however, possibly involved two of these categories, depending on interpretation. Figure 5.5 provides an example involving use of *autosum* where the default range was selected, the spreadsheet function was used incorrectly, but it generated an algorithm error because the

range was wrong and possibly included cells that should are not part of the problem. But the error was caused by an incorrect use of a spreadsheet function.

	A	B	C	D	
1		**Pennies**	**Nickels**	**Dimes**	
	Michelle	3	2	3	=SUM(B2:D2)
	Tricia	3	1	4	=SUM(B3:D3)
	Vicky	2	4	2	=SUM(B4:D4)
		=SUM(D5:D7)	=SUM(E5:E7)	=SUM(F5:F7)	=SUM(B5:D5)

Incorrect algorithm generate by *autosum*

Figure 5.5 Error that is difficult to categorize using the reasoning categories of the taxonomy developed by Rajalingham et al. (2001)

Another example was in the calculation of monthly payment, a common error was to not convert Annual Percentage Rate (APR) to monthly. This could be classified as a real-world error or a spreadsheet error, depending on the background of the developer.

Most previous studies were based on a set of errors in spreadsheet development at one particular time. The current study added the dimension of learning over time in a spreadsheet course. A notable trend was that errors due to spreadsheet logic improved as the class progressed. Skills learned early on in the course improved with practice and participants in the study and the class as a whole showed improvement on spreadsheet items such as referencing techniques and using simple statistical functions. Thre has been some evidence that spreadsheet accuracy may improve in realtion to the amount of tim e of instruction (Kruck et al., 2003). The exception to this trend was Participant B

continuing to enter individual formulas rather than using absolute referencing or an IF() function. Students struggled with many of the mathematical concepts and their integration into spreadsheet formulas, although likely some transfer of mathematical reasoning was a problem if the mathematical components of a problem were similar to previous problems in the class.

In this study a few instances in error-making existed in which a term was left out of a formula or a conversion forgotten. The taxonomies used for spreadsheet error classification included a category for *omission* errors. However, the taxonomy omission errors were based on the scope of the model, rather than on individual cells. If the participants left out a part of the model (i.e. benefit deduction in payroll calculations), it was considered an omission error. In Panko and Sprague (1999), the researchers felt that the omission errors might be attributed to the task being larger than student's working memory, which must hold planning information as well as data. Kintsch (1998) and Hendry and Green (1994) suggested that interacting with the spreadsheet added to cognitive load and complicated the problem-solving process**.**

In the current study, the cases where factors were left out of formulas (i.e. scaling errors) and parameters were left out of spreadsheet functions needed to be categorized based on the taxonomy. If a factor was left out of a formula, it was considered a real-world (algorithm) error, in that an incorrect algorithm was chosen. There could be a question as to whether the student did not know that correct algorithm, was not able to develop one, or lost some of the factors while working memory was engaged in the problem. If a student left out a parameter of a spreadsheet function (IF, PMT), it was

classified as a spreadsheet logic error, even though the student may have remembered at some point that that function required additional parameters. In many cases, the function still produced a value and might even have produced a correct value for a specific case.

When solving problems one of the reasoning approaches used was analogical reasoning. In the present study, the participants tried to adapt ideas and concepts learned in class to the spreadsheet tasks. Some people have argued that this action may not be analogical reasoning, but case-based learning or a form of mimicry (Lithner, 2000). Students applied simplified strategies using superficial reasoning, basing their strategy on identifying similarities and mimicking procedures instead of understanding fundamental mathematical concepts. This action seemed useful for spreadsheet skills and reasoning, but did not help with mathematical reasoning. They did at times seem to "remember" that scaling was needed for a certain spreadsheet function, but did not "reason" that scaling was necessary. The participants were able to develop formulas that generated a value in the spreadsheet, although the value may have been incorrect. They had some problems with spreadsheet functionality such as filling and copying, but could either figure it out or use a workaround. Moreover, as the course progressed and students had more practice with spreadsheet skills, this information became more intuitive and autonomous. This activity could sometimes be a detriment to spreadsheet accuracy if students did not learn accurately when, where, and how to use techniques and evaluate results.

Emergent Themes

Four notable themes emerged during the study involving student reasoning during spreadsheet application development and the subsequent analysis of data. First, students may use different reasoning skills and approaches during the mathematical phases of spreadsheet development. During mathematical reasoning, they reverted to word problem solving type reasoning, a painful combination of trial-and-error, guessing, and mimicry. Spreadsheet reasoning seemed to be more like solving a puzzle: filling in the cells. Bishop-Clark (1995) found that while it was assumed that a particular type of reasoning was used during computer programming, actually different types of reasoning were being used during the phase of development (design, coding, debugging). The mathematical reasoning used by participant in the study was prone to errors and did not improve through additional time learning spreadsheet reasoning. In addition, the participants used alternative methods to accomplish a given spreadsheet task. If they were unable to remember or implement what they thought was the most effective method, they resorted to reasonable alternatives. In the mathematical phase of reasoning, the participants seemed to think there was only one way to solve the problem, and often they did not know that one way.

The second idea that emerged as the research progressed involved the conceptualization of mathematical functions with a spreadsheet toolbar icon. In particular, the aggregate statistical functions (sum, max, min, average) were a click of the *autosum* icon ($\sum$) on the Excel toolbar, rather than a measurement of data tendencies. This phenomenon had two possible implications for error-making. The default selection

of the *autosum* function may have been incorrect. If the students were not connecting the process with the mathematical concept, they were not able to effectively evaluate the result. The participants used function buttons to calculate average, minimum, and maximum. Very little, if any, critical thinking was used on the result. They seemed happy that a value resulted and they could copy or fill the formula to other cells.

One phenomenon was observed in the creation of the same error for all three participants on the practice activity. They were presented with numbers of coins and asked to develop a formula to compute the amount of money the collection of coins represented. All participants used an *autosum* function to get the total number of coins rather than the total value of the coins. The implementation of this method may have been due to their interpretation of the problem, rather than due to improper use of the spreadsheet function. At that point in the course, the students had encountered several situations requiring columns or rows of data to be summed or averaged. Based on the students' recent experiences, an automatic reaction could have been to use *autosum*. The error would then be due to fault analogical reasoning, as observed by Anderson (2000).

The third emergent concept concerns the reasoning used during formula copy and fill operations. The student thought process did not appear to have included mathematical or algorithmic components. The student spreadsheet developers were more-or-less "painting" one formula into multiple cells. If they believed the computer and did not critically assess the results, then no mathematical reasoning was taking place.

This study is perhaps the first attempt to categorize errors based on the reasoning categories in the taxonomy developed by Rajalingham et al. (2001). This taxonomy was

revised by Rajalingham (2005) and later by Pursar and Chadwick (2006) but the reasoning categories have remained the same. Most previous studies used a taxonomy that is better suited for spreadsheet errors in existing spreadsheets rather than examination of the source of errors in spreadsheets under development. The fourth emergent theme involved the effectiveness of the taxonomy by Rajalingham et al. (2001) for categorizing spreadsheet reasoning errors. The goal of taxonomy is to classify errors based on characteristics with minimal overlap between categories. In this study using the Rajalingham et al. (2001) taxonomy, there were several instances where one error might be interpreted to be in multiple categories. Also, errors from multiple categories might occur within one cell. The categories implied separate phases of formula development: algorithm selection, mathematical representation, and spreadsheet implementation. In reality, the participants used an iterative development process that jumped among these processes, often with errors being made and corrected during development.

Limitations

The study had several limitations due to the nature of the investigation, size and background of the participant group, and size and scope of the spreadsheets. The observation and analysis of the reasoning and the learning processes of three participants developing spreadsheet applications do provide some insight into student reasoning during spreadsheet development but cannot be generalized to all spreadsheet developers. As the participants were novices with respect to spreadsheet development in an

introductory spreadsheet course, the results do not reflect the reasoning or development processes of more experienced spreadsheet developers in a real-life setting.

The literature (Powell et al., 2008) notes that lab-based studies involve small spreadsheets developed by a single user in a relatively short period of time and may not be used again. Operational spreadsheets are developed by a team over a period of time and are typically much larger than those developed in lab-based situations. The spreadsheets used in the current study involved a small number of cells and worksheets. Most of the data elements were visible on the computer screen at any one time, thus the effects of an action on one cell on other cells were visible to the developer. This situation is not usually the case in an operational spreadsheet used in industry, business, or research. Thus, the effects of cognitive overhead mentioned in this study due to keeping track of screen elements was less than that for larger, more complex spreadsheets. However, since the students were initially learning these techniques, skills and concepts, the learning processes added to the normal spreadsheet development cognitive load.

The tasks used in the study and in the course associated with the study were developed to use little specific domain knowledge for a particular field. In a business, industry, or research setting, the spreadsheet developer more likely deals with familiar domain-specific concepts as well as mathematics concepts and spreadsheet capabilities. This recognition means there should have been few real-world knowledge errors in the current study. An operational spreadsheet used in industry, business, or research is more prone to real-world errors, however, in the development of an such an operational

spreadsheet, the assumption is that domain-specific knowledge is available and used in development.

Since the study was a lab-study, explicit directions were given for some parts of the tasks. Providing direct guidance was intended to focus student attention on the analytical portion of a spreadsheet application. The directions and templates may have affected the quantity and types of errors compared to those generated if the spreadsheet was developed from scratch and formulas were not provided. Also, since the participants were students in a class taught by the researcher/instructor, intervention may have affected development process.

A five-week spreadsheet course provided foundation with a basic set of skills and concepts. However, fluency in spreadsheet application development requires much more experience. Powell et al. (2008) reported that only half of spreadsheet developers had formal training. Thus, even with the short introduction to spreadsheets, these students may have represented the type of work of about half of the spreadsheet developers.

Future Directions

The results of the current study suggest several areas for future research. The effect of mathematical knowledge prior to learning about spreadsheets, the persistence of the spreadsheet knowledge over time, the reasoning used during *autosum* and fill/copy operations, the impact of design on spreadsheet accuracy, and the effect of knowledge about spreadsheet errors on accuracy may all benefit from further investigation. As an alternative to the think-aloud protocols, the process of analysis of student thinking might

incorporate work with pairs of students. Another useful study might be to conduct a study similar to this study using experienced spreadsheet developers. Benefits may be derived from analyzing the effect of introducing instruction on error-making to determine if errors are reduced. The following paragraphs describe these avenues of potential research.

To determine the basis for mathematical reasoning errors, an investigation is needed of whether the students' difficulty is in transferring existing knowledge to a spreadsheet environment, or that they do not fully understand the mathematical concepts involved. A mathematics assessment would help determine if the students are able to solve similar problems without a spreadsheet. If they are able to solve the problems mathematically, then transfer to the spreadsheet context may be the problem. If the students cannot solve the problems mathematically, then they may not have achieved an understanding of the mathematical concepts necessary to use the ideas beyond what was required to pass a mathematics course. Certainly errors result if the student's mathematical reasoning is flawed from the beginning.

During the study, the students appeared more confident and adept with spreadsheet skills and concepts than mathematical skills and concepts. However, they actively participated in the spreadsheet course at the time. Examining the same students after a period of time might help determine if they actually understand the spreadsheet concepts or knew enough to pass the course. Spreadsheet logic errors may result in later spreadsheet development if concepts are not understood.

Student perception and reasoning around the use of the *autosum* feature and the copy/fill operation need to be investigated. Helpful information would come from the consideration of how student spreadsheet developers think of the arithmetic operation or statistical measurement being implemented or whether they are simply clicking on an icon. Wilson, Ainley, and Bills (2003) noted that cognitive processes used during the copy/fill operation should be researched. In the current study, it appeared that students treated the copy/fill/drag operation almost as "painting" in a graphical design application. They did not seem to recognize the notion of iterating an operation over a set of values. If they are not reflecting on the operation they are performing and the results, errors can result from accepting default ranges or using improper referencing techniques. If students are implementing a function that performs operations on a number of cells, they should be cognizant of the operation they are using and the results, otherwise the process increases the likelihood of spreadsheet logic errors.

In the setting provided, and many of those in the research literature, developers are given a problem to implement in a spreadsheet on the spot. While that development process seems to fit the spreadsheet model of ad-hoc development, it would not be the approach suggested based on software engineering methodology. Janvrin and Morrsion (2000) investigated the impact of design in spreadsheets using data flow diagrams, but other design methods may be effective. It might be beneficial to learn if there is an impact on errors if developers are engaged in a formal design process prior to being engaged in the spreadsheet development. Developers could study the problem prior to spreadsheet development where they think about spreadsheet formulas and functions

needed and generate relevant questions. They could then develop a prototype layout and design, and a set of formulas to be used prior to beginning the implementation process.

Observing and recording the verbal interactions of pairs of students developing spreadsheets could provide some insights into the reasoning being used. The discussion between students might include strategy selection, error correction, and comparison of reasoning approaches. Questions from the researcher could help obtain information, if the conversation did not include it. An interview afterward could probe for areas of reasoning. Panko and Halverson (1997) tested the effect of groups in detecting errors, but did not investigate the reasoning involved during development. Wilson, Ainley, and Bills (2003) used this methodology with elementary school students solving problems using mathematics alone and using a spreadsheet, but did not focus on error-making.

One goal in education is to impart knowledge to learners that has been developed over a long period of time based on experience. Observing experienced spreadsheet developers as they complete the same tasks used in this study would allow for a comparison of the strategies and reasoning used against those of the students who were novice spreadsheet developers. Panko and Sprague (1998) compared the error rates and types of errors created by experienced and novice spreadsheet developers. However, they used a taxonomy that categorized errors as mechanical, omission, or logical and did not account for reasoning that led to the errors. They found that experienced spreadsheet users had a similar number of errors and a variety of errors similar to novices. However, there are likely reasoning differences between the experienced users and novices.

Knowledge of these reasoning differences may help determine how to impart expert reasoning to novices.

One approach to reducing spreadsheet errors is to educate users about errors and their impact on the accuracy of spreadsheets. It would be useful to investigate the effect of including content and activities about spreadsheet errors in a course on the accuracy of the spreadsheets developed. Chadwick and Sue (2001) added instruction on error-making to spreadsheet instruction. However, they did not investigate the number and types of errors generated after students had the instruction. If error rates were reduced through this type of instruction, it could suggest adding a component on errors to spreadsheet instruction.

Suggestions for Improvement to Spreadsheet Instruction and Learning

Based on the results of the analysis of the data from think-aloud sessions and interviews in the current study, as well as the classroom observations, some suggestions can be put forth for improving spreadsheet education to reduce errors.

First, spreadsheet education is essential. Little research exists on how to prevent spreadsheet errors, but certainly, education is one avenue. Colleges are curtailing course offerings as they are assuming that students already know about spreadsheets from high school or learn it in other courses. As noted by Baker et al. (2005), only half of spreadsheet users/developers receive formal training. People learn about spreadsheets through a variety of means, for various lengths of time. It is clear that after a five-week introduction to spreadsheet course, the students have progressed beyond the novice stage

in terms of information they know about spreadsheet capabilities. They, however, have had little practice in developing real-life spreadsheets. Some investigation into the effect of longer term instruction on error rates does exist (Kruck et al., 2003), but this area needs to be investigated more thoroughly.

Adding design, testing, and debugging components to spreadsheet education as well as information about the impact and causes of spreadsheet errors may help increase accuracy during spreadsheet development. These phases of development have been part of software engineering for decades. Spreadsheet development includes standard results from spreadsheets given nonstandard development practices. Although the addition of design, testing, and debugging takes away from some of the benefits of spreadsheets for ad-hoc development, the resulting product may be more accurate. Introducing the use of test data or test variables could help developers think in terms of accuracy of the results. Several researchers have investigated the effects of design and testing on error-rates of spreadsheets, but few have suggested when, where, or how those elements should be taught. As students seem to view filling the cells in the spreadsheet as a puzzle, perhaps debugging and testing could also be viewed in this manner, providing motivation for learning about errors. Connecting the accuracy of a spreadsheet to real-life implications of errors might also increase motivation for learning.

The third suggestion for improving spreadsheet instruction and learning is that spreadsheet and mathematics curriculums should be more fully integrated. In the current study, the participants' problems with mathematical pieces of development were attributed to either difficulties in transfer of mathematical knowledge to a different

domain or lack of mathematical knowledge. If the former is the case, then integration of the two domains would introduce students to an environment in which they solve mathematical problems using spreadsheets at an earlier point in their education than college. If the second possibility is the case, that the student never fully understood the mathematical concepts enough to use them, then perhaps integrating another learning environment might have a positive effect on learning. Spreadsheets have been available for use in mathematics course for a number of years, yet few of the students in this college course were very familiar with the spreadsheet environment other than as an end-user. Since students enjoy working with spreadsheets, using the spreadsheet in mathematics courses may increase motivation to learn. Certainly many of the current and future careers rely on expertise with the integrating mathematical ideas into a spreadsheet environment.

Finally, although the spreadsheet can be used as a cognitive tool in problem solving and learning, instruction and learning are often skills-based. That is, in a formal course, individuals are usually taught a set of spreadsheet skills, operations, and functions in a limited set of contexts. Spreadsheet reasoning should be taught and learned based on the premise of the spreadsheet as a cognitive tool used for problem-solving rather than as a skill-set. The cognitive tool allows the developer/learner and technology to work in a partnership where the technology eases some of the tedious cognitive burden and provides a platform for representation of ideas, testing solution methods, reflection on reasoning, analysis of outcomes, critical thinking, integrating success and errors into

learning and knowledge, effective iterative revision of problem and solution, and meaningful learning.

The spreadsheet should help a developer or learner understand the problem and solution as well as associated reasoning. In an educational setting, the learner should begin to reason about the selection between alternative solutions. Spreadsheets provide the capability to help understand algebraic and mathematical principles, graphing and graph analysis, simulation and modeling, as well as domain-specific content. In addition, the learner should gain knowledge that can be used to continue to learn about spreadsheets and their capabilities. Learn how to learn and the reasoning application

The resulting products of spreadsheet development are applications that are used for meaningful purposes. Often, in spreadsheet classes, an application is developed but hardly used. The student does not an opportunity to understand the implication of localized development for specific situations. Applications should be used and revised during the course of instruction providing learners incentive and experience to design more reliable and accurate applications.

This suggestion combines many aspects of the previous three. In order to learn how to effectively use the spreadsheet as a cognitive tool for reasoning and problem solving, a substantial instruction/learning period is necessary. Also, developers need to learn to reason mathematically using a spreadsheet. Design, testing, and debugging are normal components of problem solving, reasoning, and software development. The integration of these components into spreadsheet education requires that students learn how to think about the problem prior to implementation, decide what reasonable test data

and results are, find errors, and determine the most effective method for resolving the error.

Comment [M1]: not a sentence .. I think?

The spreadsheet tool does not provide them with an automatic avenue to initiate student mathematical reasoning and problem solving. One drawback of use of technology in learning is that the focus can drift to the technology itself, rather than being integrated with the problem solution. Another drawback is the tendency to believe the computer output without critically evaluating the results. Learning needs to be facilitated to ensure that this technology is used effectively and appropriately. Based on cognitive theory, for effective and meaningful learning, learners must be engaged in the learning process, believe that they are making choices about their learning, and reflect on their learning (Mayer, 2002).

References

Abraham, R. and Erwig, M. (2006). Inferring Templates from Spreadsheets *28th IEEE/ACM Int. Conf. on Software Engineering*, 182-191.

Anderson, J.R. (1989). The analogical origins of errors in problem solving, In Klahr, D. and Kotovsky, K. (Eds.). *Complex Information Processing*, Lawrence Erlbaum Associates, Hillsdale, N.J.

Anderson, L. W. & Krathworhl, D. R. (2001). *A taxonomy for learning, teaching, and assessing: A revision of Bloom 's taxonomy of educational objectives*. NY: Addison Wesley Longman, Inc.

Baker, K.R., Powell, S.G., Lawson, L., & Foster-Johnson, L (2005). A survey of spreadsheet users. presented at the Spreadsheet Engineering Project Institute 2005, San Francisco.

Bernardo, A.B.I. (2001). Analogical problem construction and transfer in mathematical problem solving. *Educational Psychology*, 21, 2, 34-38.

Bishop-Clark, C. (1995). Cognitive style, personality, and computer programming, *Computers in Human Behavior*, 11(2), 241-260.

Bransford, J. D., Brown, A. L., & Cocking, R. R. (Eds.). (1999). *How People Learn: Brain, Mind, Experience, and School*. Washington, D.C.: National Academy Press.

Brown, P. S., & Gould, J. D. (1987). An experimental study of people creating spreadsheets. *ACM Transactions on Office Information Systems, 5(3),* 258-272.

Burnett, M., Cook, C., Pendse, O., Rothermel, G., Summet, J., & Wallace, C. (2003). End-user software engineering with assertions in the spreadsheet paradigm. *Proceedings of the 25th international conference on Software engineering*. Portland, Oregon.

Chadwick, D. & Sue, R. (2001). Teaching spreadsheet development using peer audit and self-audit methods for reducing errors. *Proceedings of the European Symposium on Spreadsheet Risks*. Amsterdam, July.

Charney, D., Reder, L., & Kusbit. G. (1990). Goal setting and procedure selection in acquiring computer skills: A comparison of tutorials, problem solving, and learner exploration, *Cognition and Instruction*, 7, 323-342.

Clarke T., Ayres P., & Sweller, J. (2005). The impact of sequencing and prior knowledge on learning mathematics though spreadsheet applications. *Educational Technology, Research and Development*, 53(3), 15-24.

Clermont, M. (2003). A scalable approach to spreadsheet visualization, Dissertation, Universitat Klagenfurt, Austria.

Cook, J.L. (2003). College Students and Algebra Story Problems: Strategies for Identifying Relevant Information, *Reading Psychology*, 27(2-3), 95-125

Feicht, L. (2000). Developing Algebraic Thinking By Using A Spreadsheet Approach To Algebra Word Problems, *Learning and Leading with Technology*

Galletta, D.F., Hartzell, K.S., Johnson, S.E., Joseph, J.L., Rustagi, S.(1997). Spreadsheet presentation and error detection: An Experimental Study. *Journal of Management Information Systems, 3, 45-63*

Gick, M.L. (1986). Problem-solving strategies. *Educational Psychologist*, 21, 99-120.

Grossman, T. (2002). Spreadsheet Engineering: A Research Framework, *Proceedings of the European Symposium on Spreadsheet Risks*. Cardiff,, July

Ericcson, K.A. & Simon, H.A. (1993). *Protocol Analysis: Verbal reports as data.* Cambridge, Ma: The MIT Press.

Hendry, D.G., & Green, T.R.G. (1994). Creating, comprehending and explaining spreadsheets: a cognitive interpretation of what discretionary users think of the spreadsheet model. *International Journal of Man-Machine Studies, 40*, 1033-1065.

Howe, H & Simkin, M.G. (2006). Factors Affecting the Ability to Detect Spreadsheet Errors, *Decision Sciences Journal of Innovative Education*, 4(1), 101-122.

Isakowitz T., Schocken S., & Lucas, H.C. (1995). Toward a logical/physical theory of spreadsheet modeling. *ACM Transactions on Information Systems*, 13(1),1–37.

Janvrin, D. & Morrison, J. (2000). Using a structured design approach to reduce risks in end user spreadsheet development. *Information & Management*, 37, 1-12.

Jonassen, D.H. (1995). Computers as cognitive tools. *Journal of Computing in Higher Education*, 6(2), 50-54.

Kintsch, W. (1998). *Comprehension: A paradigm for cognition*. Cambridge: Cambridge University Press.

Koedinger, K. R. & Nathan, M. J. (2004). The real story behind story problems: Effects of representations on quantitative reasoning. *Journal of the Learning Sciences*, 13(2), 129-164.

Kolodner, J.L., & Guzdial, M. (1999). Theory and practice of case-based learning aids. In Jonassen, D.H. & Land, S.M. (Eds.*), Theoretical Foundations of Learning Environments*. Lawrence Erlbaum Associates: Mahwah, NJ.

Krathwohl, D.R. (1998). *Methods of Educational and Social Science Research: An Integrated Approach.* Waveland Press.

Kruck, S.E., Maher, J.J., & Barkhi, R. (2003). A framework for cognitive skill acquisition and spreadsheet training in end-users, *Advanced Topics in End User Computing*, Idea Group Publishing, Hershey, Pa., 212-233.

Lambert, N.M., & McCombs, B.L. (Eds.) (1998). *How Students Learn: Reforming Schools through Learner-Centered Instruction.* Washington: American Psychological Association.

Lithner, J. (2000). Mathematical reasoning in task solving. *Educational Studies in Mathematics*. 41, 165-190.

Mayer, R.E. (2002). *The promise of educational psychology, volume II: Teaching for Meaningful Learning*. Merrill Prentice Hall.

Nardi, B.A. & Miller, J. R. (1990). An ethnographic study of distributed problem solving in spreadsheet development. Proceedings from CSCW'90: *Computer Supported Cooperative Work*. Los Angeles, CA. 197-208.

Nardi, B.A. (1993), *A small matter of programming: Perspectives on end user programming*, M.I.T. Press.

Ozgun-Koca, S. 2000. Using Spreadsheets in Mathematics Education. *ERIC Educational Reports.* December, 2000.

Panko, R. (2000), Two corpuses of spreadsheet errors. System Sciences 2000: *Proceedings of the 33th International Conference on Systems Science. Honolulu, HI.*

Panko, R. & Halverson, R.P, Jr. (1997). Are two heads better than one (at reducing spreadsheet errors in spreadsheet modeling?) *Office Systems Research Journal* 15(1), 21-32.

Panko, R. (2005). What we know about spreadsheet errors, *Human Error Website*, Honolulu, HI: University of Hawai`i. Retrieved May 15, 2005 from http://panko.cba.hawaii.edu/HumanErr/.

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

Panko, R. and Halverson, Jr., R. P., 1997, Are two *heads* better than one? (at reducing errors in *spreadsheet* modeling). *Office Systems Research Journal* (15:1), 21-32.

Panko, R. and Sprague, R.H. Jr. (1999). Hitting the wall: Errors in developing and code inspecting a 'simple' spreadsheet model. *Decision Support Systems*, 22(4), 337-353.

Patton, M.Q. (2001). *Qualitative research and evaluation methods*. Thousand Oaks, CA: Sage Publications, Inc.

Peyton Jones, S.P., Blackwell, A., & Burnett, M., (2003). A user-centered approach to functions in excel, *Proceedings of the ACM International Conference on Functional Programming*, Uppsala, Sweden.

Powell, S.G., Baker, K.R., & Lawson, B. (2008). A Critical Review of the Literature on Spreadsheet Errors, *Decision Support Systems*, 2008.

Purser, M.,Chadwick, D. (2006) , Does an awareness of differing types of spreadsheet errors aid end-users in identifying spreadsheet errors?, *Proceedings of the European Spreadsheet Risk Interest Group Annual Conference*, Cambridge, UK (2006) 185–204.

Rajalingham, K. (2005), *A Revised Classification of Spreadsheet Errors*, EuSpRIG Conference Proceedings, University of Greenwich, July 7-8.

Rajalingham, K., Chadwick, D.R., & Knight, B. (2001). Classification of spreadsheet errors, British Computer Society (BCS) *Computer Audit Specialist Group (CASG) Journal*, 10 (4), 5–10.

Reason, J. (1990). *Human Error*. New York, NY: Cambridge University Press.

Reimann, P. & Schult, T.J. (1996). Turning examples into cases: Acquiring knowledge structures for analogical problem solving. *Educational Psychologist*, 31(2), 123-10.

Reimann, P. & Neubert, C. (2000). The role of self-explanation in learning to use a spreadsheet thorough examples. *Journal of Computer Assisted Learning*, 16, 316-325.

Robertson, S.I. (1997). Is analogical problem solving always analogical? The case for imitation. HCRL Technical Report. Human Cognition Research Laboratory, The Open University.

Robins, A., Rountree, J., Rountree, N. (2003). Learning and teaching programming: A review and discussion. *Computer Science Education*, 13(2), 137-172.

Rothermel, K. J., Cook, C. R., Burnett, M. M., Schonfeld, J., Green, T. R. G., & Rothermel, G. (2000). WYSIWYT testing in the spreadsheet paradigm: An empirical evaluation. *Proceedings of the 22nd International Conference on Software Engineering*, 230-239.

Rutledge, J. and Rhea, V., (1991). Principle investigators report: The application of electronic spreadsheets across the curriculum. In Goodson, I. Mangan, M. & Rhea, V. (Eds.), *Curriculum and context in the use of computers for classroom learning: Interim Report #5 (pp.* 213-227). London, Ontario.

Sajaniemi, J. and Tukiainen, M. (1996). Goals and plans in spreadsheets and other programming tools. In P. Vanneste, K. Bertels, B. De Decker, J-M. Jaques (Eds.), *PPIG 8, Proceedings of the 8th Annual Workshop of the Psychology of Programming Interest Group* (pp 114-122), Belgium.

Scheiter, K. and Gerjets, P. (2002). The impact of problem order: Sequencing problems as a strategy for improving ones performance. *24th Annual Conference of the Cognitive Science Society*. Fairfax, Va.

Sheehan, J. and Tessmer, M. (1997). A construct validation of the mental models learning outcome using exploratory factor analysis. *Proceedings of Selected Research and Development Presentations at the 1997 National Convention of the Association for Educational Communications and Technology*, Albuquerque, (pp 363 -380)

Stevens, S.P. and Palocsay, S.W. (2004). A translation approach to teaching linear program formulation. *Informs: Transactions on Education*, 4(3).

Tukiainen, M. (2000) Uncovering effects of programming paradigms, errors in two spreadsheet systems. *12th Workshop of the Psychology of Programming Interest Group*, Cozenza Italy, April 2000.

Wallace, C., Cook, C., Summet, J., & Burnett, M. (2002). Assertions in end-user software engineering: A think-aloud study". *IEEE Symposia on Human-Centric Languages and Environments (HCC'02)*, 63.

Whittle, D. & Janvrin, D. (2002), Do cell labels improve spreadsheet outcomes? *American Accounting Association Accounting, Behavior and Organization MidYear meeting*, October.

Wilson, Ainley, and Bills (2004). Spreadsheet generalising and paper and pencil generalizing. Proceedings of the 28th conference of the International Group for the Psychology of Mathematics Education, v4, 441-448

Wong, A. (2003). Before and beyond systems: An empirical modeling approach. PhD thesis, Department of Computer Science, University of Warwick, UK

APPENDICES

Appendix A

Spreadsheet Tasks for Think-Aloud Sessions

A Practice Task

Problem

Determine the total value of the coins these people have in their pockets and find the total value of the types of coins (pennies, nickels, dimes, quarters).

Instructions

Lauren, Kristen, and Ryan all have money in their pockets. The amount of each type of coin they have is given for quantities in the spreadsheet template. Your task is to complete the spreadsheet so that it finds out how much money each has in coins and what the cumulative value is of each coin.

	A	B	C	D	E	F
1						
2						
3	**Coins**	**Pennies**	**Nickels**	**Dimes**	**Quarters**	**Totals**
4	**Person A**	2	4	2	2	
5	**Person B**	3	2	1	2	
6	**Person C**	4	4	2	3	
7	**Grand Total**					

- Compute the total value for each persons set of coins in column F.
- Compute the total value of each coin in row 7.
- Compute the grand total of all the money the three people have in cell F7.

Format the currency result with 2 decimal places.

Remember to talk out loud, explaining your thinking as you complete this spreadsheet.

Task Examples

Following are potential tasks indicating the level of difficulty and complexity of each session spreadsheet. To reduce cognitive load and stress, templates will be provided for the second and third tasks. This reduces development time, cognitive load, and frustration and allows the investigation to focus on common error situations.

Spreadsheet Application Task 1

The Wall

The *wall* is a task developed by Panko to be a relatively simple, domain-free application. It has been used in several research investigations (Panko, 2001; Burnett et al., 2003)

Problem

You are to build a spreadsheet model to help you create a bid to build a wall. You will offer two options, - rock or brick The wall will be built by a crew of two, working 3 8-hours days. The wall will be 20 ft long, 6 ft tall, and 2 feet thick. Wages are $10/hour/person. You will have to add 20% to wages to cover fringe benefits. Brick costs $2 per cubic foot. Rock costs $3 per cubic foot. Your bid must include a profit margin of 20% beyond your expected cost.

Instructions

Keep in mind when you construct this spreadsheet application that

- The spreadsheet should be readable. Someone else could look at it and figure out what it's about. You should be able to look at it next week and figure it out.
- The spreadsheet should be flexible
- Some information is variable and may change later, (i.e. pay rate, cost per cubic foot, etc). It would be advantageous to be able to just change 1 number and have the calculations automatically recalculate the new result, rather than having to make adjustments to formulas.

Remember to talk out loud, explaining your thinking as you complete this spreadsheet.

Spreadsheet Application Task 2

Problem

The mileage spreadsheet contains mileage data for several makes and models of vehicles. You are to develop the rest of the application that computes several values based on fuel prices, fuel tank capacity, and mileage.

Instructions

Use **absolute reference/range names** and **fill/copy** operations wherever it is effective. If a **fill/copy** does not work the way you think it should, you may have to enter the formula for each cell separately.

Remember to talk out loud, explaining your thinking as you complete this spreadsheet.

1. Enter the formulas in the first row for the following
   - In cells F7, the average MPG in based on the min and max values in the two previous columns using the appropriate function
   - In cells G7 cost of driving a 200 mile trip based on average mpg and gas price in cell G2
     Cost = 200/average mpg * Gas Price
   - In cells H7 calculate the amount of EPA discount
     The Environmental Protection Agency offers a 5% discount in gas to vehicles with high mileage rates (this is hypothetical).
     If the average mpg is greater than 25 there is a 5% discount on the Cost of going 200 miles. Create a formula to calculate the amount of the discount. You should use an **IF** function.
   - In cells I7 enter the formula to calculate adjusted cost for the EPA discount Cost of driving 200 miles – EPA discount.
   - In cells J7 enter a formula to calculate the distance that can be driven on a tank of gas, based on the average mpg and tank size.

   a. In cells K7 enter a formula to determine the cost of a tank of gas
   b. You want to determine the effect of increases on the cost of a tank of gas.
      In cell L7 enter the formula to calculate the effect of the % increase in cell L2 on the value in cell K7

2. Fill down the formulas, adjusting as necessary

3. In row 15 and 16 enter functions to calculate min and max of each column to the right.

4. Modify values

a. Change the 10% increase in cell L2 to 4%
b. Change the price of gas in cell H2 to be $2.89

						$/gal					
						2.87					5%
Make	**Vehicle**	**tank size**	**MPG City**	**MPG Hiway**	**Avg MPG**	**Cost of going 200 miles**	**EPA dis-count**	**Adj. Cost**	**Dist. on one tank**	**Cost of a tank of gas**	**Cost with increase**
Ford	**Escape**	**16.5**	**19**	**22**							
Ford	**Focus**	**14**	**26**	**32**							
Subaru	**Legacy**	**16.9**	**19**	**25**							
Dodge	**Stratus**	**16**	**22**	**30**							
Honda	**Civic**	**13.2**	**32**	**37**							
Honda	**Hybrid**	**11.9**	**47**	**48**							
Subaru	**Forester**	15.9	20	23							
Toyota	**Forerunner**	23	17	21							
	Min										
	Max										

Task2 : Starting Template

Remember to talk out loud, explaining your thinking as you complete this spreadsheet.

Spreadsheet Application Task 3

Income/Expense

Problem

This task involves determining the income, expenses, and profit for a hypothetical computer company. The application, when finished, should allow for input values (the ones currently given) to be changed, with the spreadsheet accurately reflecting the changes.

Instructions

This workbook contains two worksheets, computer sales and payroll. The computer sales spreadsheet has 5 sections: income, mortgage, expenses, and profit

Enter the formula and functions necessary to complete the calculations in each section based on the information below.

Remember to talk out loud, explaining your thinking as you complete this spreadsheet.

Payroll

The payroll data is on the worksheet labeled pay. The employee's hours may change each week so the application should be flexible enough to respond to changes. The resulting application should also be able to have more employees entered in the rows below,

- Add any columns you need to calculate the information the way you understand it.
- Format cells appropriately.
- Hours over 40 are considered overtime and pay is time and a half.
- tax is based on the Gross pay and the tax rate in cell H1
- If an employee works over 20 hours in a week, they also pay 10% of the Gross pay in other deductions. If the employees work less than 20 hours a week, they pay no deductions for that week.
- Calculate net pay based on gross pay, taxes, and deductions.
- In row 7, enter totals for hours worked, regular pay, overtime pay, taxes, deductions, net pay.

Income

- Income per model in cells D2 to D4
- Totals in row 11

- Percent of total income (in cell E11) for each model
- Format values appropriately

Mortgage

- Compute the monthly payment in cell E15 based on the annual percentage rate (APR), number of years of the loan, and amount of the loan.
- Format values to currency with 0 decimal places

Expenses

Expenses are as follows:

- Heat $200
- Internet $80
- Telephone $75
- Payroll reference the payroll sheet. Use weekly payroll figures to estimate monthly
- Mortgage – refers to monthly payment in cell E15
- Cost of materials 60% of the total income in cell E11
- Compute the total for expenses in cell C23
- Format all values appropriately

Profit

Calculate the profit based on total income and total expenses based on the following:

- Tax Rate : if profit is greater than 25000, the tax rate is 15%, otherwise it is 12%
- Taxes based on total profit and tax rate
- Net profit. Profit after taxes.

- Change the following input values.
- Make sure the spreadsheet values reflect the changes.
- Format all values appropriately

Remember to talk out loud, explaining your thinking as you complete this spreadsheet.

Starter Templates

	A	B	C	D	E	F	G	H
1	**Weekly payroll**					8%		
2								
3	**EID**	**Name**	**Rate**	**Hours**	**Gross Pay**	**Fed tax**	**State Tax**	**Net pay**
4	**1**	Bob	$ 10.00	42				
5	**2**	Jane	$ 7.00	35				
6	**3**	Mary	$ 6.50	20				
7	**Totals**							

	A	B	C	D	E	F
1		Computer Sales				
2						
3						
4	Monthly Income					
5						
6		**Models**	**Number Sold**	**Price**	**Income per Model**	**% of Total**
7		**extreme**	13	$ 1,438		
8		**ultra**	34	$ 923		
9		**medium**	23	$ 750		
10		**cheap**	13	$ 453		
11		**Total**				
12						
13	Mortgage					
14		**APR**	**Years**	**Amount**	**Monthly Payment**	
15		6.50%	30	$300,000.00		
16						
17	Monthly Expenses					
18		Heat				
19		internet				
20		Phone				
21		payroll				
22		Mortgage				
23		Total				
24						
25	Profit					
26		Profit				
27		tax rate				
28		taxes				
29		net profit				

Appendix B

Protocol for Student Interview

Prior to the interview and during the thinking aloud observations, the researcher will note problem areas on the spreadsheet as the student works on the task. The Student Interview Protocol is designed to determine what type of reasoning used in working on the task and to assess how that reasoning developed.

Introductory Questions

1. Did you understand this activity and how to develop the application?
    - No – Which parts were difficult to understand?
2. Did the activity involve skills and knowledge you learned in the course?
    - No – What parts have not been covered in this course?
    - Yes - Did you feel you had learned the skills and concepts well enough to complete the activity?
        - No – which skills/concepts were difficult? Why
3. Is learning the skills/concepts easy or more difficult requiring more practice?

4. Did you have any problems with the spreadsheet?
    - Yes -
        - What gave you trouble?
        - Was your problem in understanding the problem and how the spreadsheet might be used to solve it?
        - What makes the spreadsheet difficult to work with?
        - Do you think you learned how to do this spreadsheet solution to the problem in the class?
        - Do you think you developed any misconceptions as you learned to create the spreadsheet to solve this problem?
    - No -
        - Were the spreadsheet skills needed in the problem covered in class?
            - Yes –
                Do you think you may have had trouble learning it?
                Is this the kind of thing that you have trouble learning?
                Why?
            - No –
                So you were ok with all the formulas?
                Yes –
                Are some harder than others?
                Were some harder to learn than others?
                Which ones and why?
                Do you feel you understand them now?

No -
Which ones were difficult?
Did you have trouble learning them initially?

5. Do you think the spreadsheet is correct?
   - Yes –
     - If you changed some of the input values, would it still be correct?
   - No –
     - Where might it be off?
     - Was it because you may have interpreted the problem, had trouble with the math or made a spreadsheet error?

Ad-Hoc questions and comments

a. Throughout , try to add comments like:
- Thanks, this is just the type of information I need.
- Thanks, that's useful information

b. Pointing to cell with error:
- There is a problem with this cell, can you see what it is?
- Do you do this same type of thing often, always?
- How did you come to think this way?
- Do you think you developed or learned this approach in the spreadsheet class, math classes, high school, someplace else?
- What problems does the error cause in the spreadsheet?

c. Pointing to cell with ineffective use of copying referencing:
- Do you feel comfortable with copying and creating references to other cells?
- Did you learn how to copy data and formulas?
- Was this difficult to learn?
- Relative and absolute referencing can be difficult concepts
  Do you feel you understand absolute and relative referencing?
- When you learned it, were there parts that didn't make sense?
- Was it easy to use in the exercises from the book?
- Do you think it's difficult to know when to use referencing in an application you are designing and developing?

c. Pointing to cell with consistent error:
There is an error that is similar to an error made previously or consistently
- Do you think this is an error you make consistently?
  Yes -
  - Is this the method you learned to do it?

Yes –

- Do you think you learned the appropriate method?
- could you have learned the method so it works sometimes and not others?

No -

- How did you figure out how to use this method?
- Are there situations where this approach doesn't give and error?
- Do you know that this is going to result in an error?

NO –

- Why does the error occur sometimes?
- Do you think maybe you learned how to do this with one type of problem and that method doesn't work with other types of problems?

Appendix C

Background Questionnaire

Name

Major

Year in School (FY, SO, JR, SR)

Why are you taking this course?
Required for major
To learn more about spreadsheets.
Other

Computer Application Background

How would you rate your computer ability in common applications (email, web browsing, word processing). Please use numbers 1-5, with 1 representing no experience and 5 representing expert.

Application	**Rating (1-5)**
Word Processing	
Web Browsing	
Spreadsheets	
Databases	
Video Games (list)	
Other (specify)	

Have you done any computer programming?
If so what language did you use?
What type of programs have you developed?

Mathematics Background

Spreadsheets involve mathematical calculations. Often one uses mathematical reasoning when developing spreadsheets. How would you rate your ability in the following mathematical areas(Circle one or provide your own terms)?

Arithmetic (excellent, pretty good, fair, struggle with, cannot do)
Algebra (excellent, pretty good, fair, struggle with, cannot do)
Word Problems (excellent, pretty good, fair, struggle with, cannot do)

What was your last math course?

When did you take that course and where.

Spreadsheet Background

Have you used a spreadsheet previously in school, at work, or in some other setting? Please describe your usage

Work

School

Other

Have you developed a spreadsheet from scratch to solve a problem (rather than using one someone else created)? If so, circle the following spreadsheet capabilities and features with which you are familiar.

Entering data
Creating simple formulas
Using statistical functions (SUM, AVG, MIN, MAX, COUNT)
Fill down, fill right
Copy
Referencing a cell in a formula
Absolute referencing
IF
PMT, FV
Pivot tables
Charts, graphs
Auditing
VLOOKUP
Sorting
Conditional formatting
Range names
Math Functions (ROUND, RAND, ABS)
Spreadsheet Subtotal Function

Appendix D

Classroom Observation Protocol

General

- date
- time
- curricular topic for the day
- attendance

Student Attitude

- Do the students seem confident of their abilities with spreadsheets?
- Do the students seem frustrated while working on spreadsheets?
- Do the students enjoy working with spreadsheets?
- Do the students seem to perceive the content as useful?
- Are the students concerned about accuracy?
- Are the students concerned about dependability of the spreadsheet?

Classroom Environment

- Are the students/class engaged in the learning process?
- Do the students ask questions?
- Do the students work collaboratively?
- Do the students finish in the allotted time?

Student understandings and misunderstandings

- Are students able to work the examples in the text?
- Are students able to apply the information to a similar activity?
- Do the students seem to have difficulties with new material?
- Do the students correct errors?
- How do the students react or respond to formative assessment, modifies reasoning, approach, or skill
- What are the student reactions, interactions, discussions on common misconceptions?
- Is there an effect of peer interaction on student reasoning?
- Are students getting help from neighbors rather than learning concepts and skills?
- Are students copying skills from the book, others or learning concepts?
- Do students ask for help with concepts or skills they should know?
- Do they test their solutions?
- What reasoning approaches are being used?
    - trial-and-error
    - means-end

  - analogical
  - case-based
  - imitation
- translating problem into logical and physical models
  - resolving problem into components determining representation in spreadsheet model
- What difficulties do students have with mathematical (real-world) concepts?
  - algebra
  - algorithm
- What difficulties do students have with spreadsheet (implementation) concepts?
  - formulas
  - referencing
  - copy/fill
  - functions
- What are student approaches used in spreadsheet application development?
  - direct
  - sequential
  - hesitant
  - confused
  - random
  - other