

DISSERTATIONS IN
**FORESTRY AND
NATURAL SCIENCES**

BENNETT FREINDERSON KANKUZI

*Deficiencies in
Spreadsheets: A Mental
Model Perspective*

PUBLICATIONS OF THE UNIVERSITY OF EASTERN FINLAND
Dissertations in Forestry and Natural Sciences No 183



UNIVERSITY OF
EASTERN FINLAND

BENNETT FREINDERSON KANKUZI

*Deficiencies in
Spreadsheets: A Mental
Model Perspective*

Publications of the University of Eastern Finland
Dissertations in Forestry and Natural Sciences
No 183

Academic Dissertation

To be presented by permission of the Faculty of Science and Forestry for public
examination in the Louhela Auditorium in Science Park Building at the
University of Eastern Finland, Joensuu, on August, 28, 2015,
at 12 o'clock noon.

School of Computing

Grano Oy

Joensuu, 2015

Editors: Research Director Pertti Pasanen, Prof. Pekka Kilpeläinen,
Prof. Kai Peiponen, Prof. Matti Vornanen

Distribution:

University of Eastern Finland Library / Sales of publications

julkaisumyynti@uef.fi

<http://www.uef.fi/kirjasto>

ISBN: 978-952-61-1827-7 (printed)

ISSNL: 1798-5668

ISSN: 1798-5668

ISBN: 978-952-61-1828-4 (pdf)

ISSNL: 1798-5668

ISSN: 1798-5676

Author's address: University of Eastern Finland
School of Computing
P.O. Box 111
80101 JOENSUU
FINLAND
email: bennetk@student.uef.fi

Supervisor: Professor Jorma Sajaniemi, Ph.Lic.
University of Eastern Finland
School of Computing
P.O. Box 111
80101 JOENSUU
FINLAND
email: jorma.sajaniemi@uef.fi

Reviewers: Professor Pertti Saariluoma, Ph.D.
University of Jyväskylä
Department of Computer Science and
Information Systems
P.O. Box 35 (Agora)
40014 JYVÄSKYLÄN YLIOPISTO
FINLAND
email: pertti.saariluoma@jyu.fi

Professor Marian Petre, Ph.D.
The Open University
Centre for Research in Computing
Walton Hall
Milton Keynes
MK7 6AA
UNITED KINGDOM
email: m.petre@open.ac.uk

Opponent: Research Fellow Sari Kujala, Ph.D.
Aalto University
School of Science, Department of Computer Science
P.O. Box 15400
00076 AALTO
FINLAND

ABSTRACT

In this thesis, we tackled the problem of errors in spreadsheets by studying spreadsheet authors' mental models. It is a common assertion that humans have mental models of the systems they interact with, and it is difficult to explain many aspects of human behaviour without resorting to a construct such as mental models. We therefore argue that it is important to first of all understand what types of mental models spreadsheet authors possess when they are doing different spreadsheet tasks in order to better understand why the spreadsheet process is so error-prone and to be able to devise new tools that better correspond to the way they think.

In the first empirical study we conducted in this research work, we investigated and characterized mental models of spreadsheet authors as they are doing various spreadsheet tasks. We found that spreadsheet authors have (at least) three mental models of a spreadsheet: the *real world model* that comprises general knowledge of the world around us; the *domain model* that represents knowledge of the problem domain and the functionality of the spreadsheet in problem domain or application terms; and the *spreadsheet model* that codes the expressions and data relationships in the spreadsheet. When explaining a spreadsheet, the real-world and domain mental models are prominent and the spreadsheet model is less evident, but when locating and fixing an error, one must constantly switch back and forth between the domain model and the spreadsheet model, which requires frequent use of the mapping between problem domain concepts and their spreadsheet model counterparts. These results suggest that a tool intended to aid in comprehension and debugging of spreadsheets should make prominent real-world and problem domain concepts and map them easily to spreadsheet-specific details.

We thus developed and evaluated a spreadsheet visualization tool that demonstrates that it is possible to devise spreadsheet authoring and debugging tools that are easy to use and that correspond to spreadsheet authors' mental models of spreadsheets

by relieving spreadsheet authors from spreadsheet details and letting them utilize more of their mental model of the application or problem domain. The tool translates traditional spreadsheet formulae into problem domain narratives and highlights referenced cells. The tool was evaluated in the second empirical study of this research work and was found to be easy to learn and helped spreadsheet authors to locate more errors in spreadsheets. Furthermore, the tool increased the use of the domain mental model when spreadsheet authors were describing errors and seemed to improve the mapping between the spreadsheet model and the domain model which is crucial in spreadsheet debugging and comprehension.

We have also put forward a case for the need to shift from the traditional spreadsheet paradigm to another paradigm in which a spreadsheet author should also be able to debug a spreadsheet in problem domain terms rather than just using traditional spreadsheet cell references. In this proposed paradigm, a spreadsheet author should also ideally be allowed to create spreadsheets by writing formulae in domain terms. We thus also developed a prototype spreadsheet visualization tool that allows spreadsheet authors to fix errors in a spreadsheet using domain terms. The tool was evaluated in the third empirical study of this research work and was found to promote spreadsheet authors to think more in domain terms in a way that overshadows the way they traditionally think when they are describing errors and even when fixing errors, hence promoting a *paradigm shift* from traditional spreadsheets.

Universal Decimal Classification: 004.67; 004.052.4; 004.416.2; 004.02; 004.832; 004.4'236; 37.015.3:159.937; 37.015.31

Library of Congress Subject Headings: Electronic spreadsheets; Human-computer interaction; Debugging in computer science; Errors; Association of ideas; Concepts - Data processing; Visual analytics - Data processing; Information visualization - Data processing; Computer-assisted instruction - Authoring programs; Decision

support systems - Software; Paradigms (Social sciences)

Yleinen suomalainen asiasanasto: taulukot; taulukkolaskenta; data; virheet; virheanalyysi; ohjelmistokehitys; visualisointi; korjaus; mielikuvat; käsiteanalyysi; paradigmmat

Acknowledgements

I thank the Almighty God for the gift of life and for the grace to undertake this research work.

My profound thanks should also go to Prof. Jorma Sajaniemi, my supervisor. I can not find words enough to express my appreciation for his wonderful supervisory skills. Indeed, I feel honoured to learn under his tutelage. The memories of the past four years, I shall always cherish.

Dr. Marja Kuittinen should also receive my special vote of thanks for her help in some aspects of the studies in the course of this work, and particularly, for her help in the validation of the data coding method in the first empirical study conducted in this research work.

I also sincerely thank the preliminary examiners of this dissertation, Prof. Pertti Saariluoma of University of Jyväskylä, Finland and Prof. Marian Petre of The Open University, United Kingdom, for their time and indeed for their valuable comments.

The University of Eastern Finland School of Computing supported me through the financial grants that supported my stays in Joensuu, Finland as well as for funding my attendance costs in international conferences in San Jose, California, USA as well as in Melbourne, Australia. I am very grateful for this.

I also sincerely thank all those who participated in the empirical studies conducted in this research work. The participants work in various companies and organizations in the city of Blantyre in Malawi, southern Africa. Their time was invaluable and I really appreciate their participation.

Special thanks should also go to Dr. Yirsaw Ayalew of University of Botswana for introducing me to the wonderful world of human-centred software engineering through spreadsheet software research in 2006 while I was doing my studies for an MSc in Computer Science there. It is a research path that I have always enjoyed.

Mr. Gift Nuka (now Dr. Gift Nuka), then our lecturer at University of Malawi (Chancellor College) during my undergraduate years from 1998 to 2002, inspired me to choose a career in Computer Science. Coming from Livingstonia Secondary School, just as many secondary school kids in Malawi that time, I had hardly heard about computers and let alone a career in Computer Science. However, the way Mr. Gift Nuka introduced computer programming to us, it really got me fascinated with programming computers. That was the genesis of my journey into the wonderful world of computing.

My wife, Cynthia and my daughter, Takondwa, have patiently waited and supported me in the course of this work. I dedicate this work to them. My late Dad, the late Freinderson Ishmael Kankuzi, inspires me, even in this research work, because he emphasized so much on the importance of education in today's world. Indeed, as Daniel Akaka, the former United States senator for Hawaii, also once said "I have witnessed how education opens doors, and I know that when sound instruction takes place, students experience the joys of new-found knowledge and the ability to excel".

Lastly but not least, I also thank my mum, Mrs Ireen Nya-Mayuni Kankuzi and every one for their encouragement and prayers.

Joensuu July 30, 2015

Bennett Freinderson Kankuzi

Contents

1	INTRODUCTION	1
1.1	Background	1
1.1.1	What are spreadsheet systems?	1
1.1.2	Importance of spreadsheets	3
1.1.3	Deficiencies in spreadsheets	4
1.1.4	Impact of deficiencies in spreadsheets	5
1.1.5	Classification of errors in spreadsheets	5
1.1.6	Spreadsheets in the context of the end-user programming paradigm	8
1.1.7	Challenges in end-user programming	9
1.1.8	From end-user programming to end-user software engineering	11
1.1.9	Proposed end-user software engineering tools in the spreadsheet context	12
1.2	Statement of the Problem	17
1.3	Research Objectives and Questions	18
1.4	Research Methodology	21
1.4.1	Justification of methodology	26
1.5	Overview of the rest of the Dissertation	28
2	LITERATURE REVIEW	29
2.1	Theory of Mental Models	29
2.1.1	Nature of mental models	29
2.1.2	Methods for “capturing” mental models	34
2.1.3	Limitations in the study of mental models	35
2.1.4	Proposed research directions in the study of mental models	36
2.2	Mental Models of Programmers	36
2.2.1	Mental models of expert programmers	37
2.2.2	Mental models of novice programmers	38

2.2.3	Techniques for characterizing programmer's mental models	40
2.3	Mental Models of Spreadsheet Users	43
2.4	Mental Model Mapping and Performance	45
2.5	Conclusion	46
3	AN EMPIRICAL STUDY OF SPREADSHEET AUTHORS' MENTAL MODELS	47
3.1	Introduction	47
3.2	The Study	49
3.2.1	Participants	49
3.2.2	Procedure	49
3.2.3	Coding of data	51
3.3	Results	52
3.4	Discussion	54
3.4.1	Existence of several mental models	54
3.4.2	Mental models between different tasks	56
3.4.3	Mental models within different tasks	57
3.4.4	Mental models in debugging episodes	59
3.4.5	Implications for tool development	63
3.4.6	Threats to validity of study	64
3.5	Conclusion	65
4	A SPREADSHEET VISUALIZATION TOOL WITH A MENTAL MODEL PERSPECTIVE	67
4.1	Introduction	67
4.2	Tool Overview	67
4.3	Label Extraction in the Tool	69
4.4	Implementation	70
4.5	Potential Uses of the Tool	73
4.6	Comparison With Other Tools	73
4.7	Conclusion	81
5	EVALUATION OF THE SPREADSHEET VISUALIZATION TOOL	83
5.1	Introduction	83

5.2	Evaluation of the Tool	84
5.2.1	Participants	85
5.2.2	Learnability	85
5.2.3	Efficiency	88
5.2.4	Satisfaction	96
5.2.5	Effect on mental models	97
5.3	Threats to Validity of Study	101
5.4	Conclusion	102
6	A CASE FOR A PARADIGM SHIFT IN SPREADSHEETS	105
6.1	Introduction	105
6.2	A Paradigm Shift in Spreadsheet Debugging	107
6.2.1	Implementation	109
6.2.2	The paradigm shift tool in action	110
6.2.3	Limitations of the paradigm shift tool	113
6.2.4	Evaluation of the paradigm shift tool	115
6.3	Conclusion	131
7	CONCLUSION	133
7.1	A Reflection on the Research Objectives	133
7.2	Our Contribution	137
7.3	Future Work	138
	BIBLIOGRAPHY	141
	APPENDICES	153

LIST OF FIGURES

1.1	An illustration of different views of a spreadsheet.	2
1.2	Steps we followed within the constructive research methodology context in this research work. Inherently are also interrelationships between the research questions.	27
3.1	A line illustration of the relative frequencies of different object reference types in the three tasks.	53
3.2	An illustration of the relationships between mental models in the explaining task and error locating task.	58
3.3	An illustration of mapping of mental models in the error fixing task.	60
4.1	Visualization of a Sales Report spreadsheet – The cell cursor is in C9, whose formula is visible at the formula bar and narrated in domain terms in the cyan coloured box. Cells referred to in the domain narrative are highlighted with cyan background. All formula cells are marked with a magenta right border.	68
4.2	An example of a domain narrative when a column name is missing. B4 is translated to “Cash” and B5 to “Inventories” and hence the formula in B6 is translated as “Cash + Inventories”.	71
4.3	An example of a domain narrative when a row name is missing. B5 is translated to “District 1” and D5 to “District 3” and hence the formula “=SUM(B5:D5)” in E5 is translated as “SUM(District 1 ... District 3)”.	71
4.4	An example of a domain narrative when both row name and column name are missing. F5 is translated to “unnamed” and F6 to “unnamed” and hence the formula “=F5+F6” in F7 is translated as “unnamed + unnamed”	71

4.5	An illustration of a high-level architecture of the domain terms spreadsheet visualization tool	72
4.6	A translation of the formula in the active cell, C9, in Spreadsheet Professional.	75
4.7	An illustration of how a symbolic name is assigned to a cell in the tool by Nardi and Serrecchia (1994) - cell C4 is referred as "PROFIT(year_2)".	76
4.8	A translation of the formula in the active cell, C9, in Spreadsheet Detective.	76
4.9	An illustration of the SUMWISE spreadsheet extension.	78
4.10	An illustration of named ranges in Numbers, the spreadsheet application.	79
5.1	A box-plot illustration for scores in the two sub-tasks.	88
5.2	Bar chart illustrating relative frequencies of errors located without the tool and with the tool for each participant.	91
5.3	Bar chart illustrating error detection rates for each participant without the tool and with the tool.	92
5.4	Bar chart illustrating relative frequencies of located errors for each error type without the tool and with the tool.	92
5.5	A line illustration of relative frequencies of overlapping object reference types in error descriptions.	98
6.1	An illustration of a high-level architecture of the paradigm shift tool.	109
6.2	An illustration of the paradigm shift tool – step 1 (an error in cell C18 whereby "Mark Watts" has been left out).	111
6.3	An illustration of the paradigm shift tool – step 2 (incorrect domain term "Jane Hill" replaced with correct domain term "Mark Watts").	112
6.4	An illustration of the paradigm shift tool – step 3 (corrected domain terms narrative exited to effect the change).	112

6.5	An illustration of the paradigm shift tool – step 4 (verification of effected error correction).	113
6.6	An illustration of the multiple occurrence label problem – step 1: An error occurs in cell E9 in that the sum is leaving out D9 or in domain terms, it is leaving out Accounting total. The correct domain narrative should be “SUM(Production Total ... Accounting Total)”. “Sales” needs to be replaced with “Accounting”.	115
6.7	An illustration of the multiple occurrence label problem – step 2: The erroneous domain narrative is corrected with “Sales” edited out and replaced with “Accounting”.	115
6.8	An illustration of the multiple occurrence label problem – step 3: The user is taken through a sequence of dialog boxes so that they choose the label that is being referred to in the domain narrative.	116
6.9	An illustration of the multiple occurrence label problem – step 4: This is the right label being referred to and as such the user answers in the affirmative to the question.	116
6.10	An illustration of the multiple occurrence label problem – step 5: The correction has been effected and the user can verify this by clicking on the formula cell they were correcting.	117
6.11	A line illustration of relative frequencies of overlapping object reference types in the error description sub-task.	121
6.12	A line illustration of relative frequencies of overlapping object reference types in the error fixing sub-task.	122
6.13	Bar chart illustrating relative frequencies of errors correctly described per error type without the tool and with the tool.	124

6.14 Bar chart illustrating relative frequencies of errors correctly fixed per error type without the tool and with the tool.	124
---	-----

LIST OF TABLES

3.1	Summary of spreadsheets and sessions in the three tasks.	52
3.2	Relative frequencies (%) of different object reference types in the three tasks.	53
3.3	Codification of nature of mental models of participants as they performed the three tasks of explaining, locating errors, and fixing located errors in their spreadsheets.	56
5.1	Error types of seeded errors and their corresponding number of seeded errors in spreadsheet S1 and spreadsheet S2.	89
5.2	Relative frequencies (%) of different object reference types in error descriptions without the tool and with the tool.	98
6.1	Error types of seeded errors and their corresponding number of seeded errors in spreadsheet S3 and spreadsheet S4.	119
6.2	Relative frequencies (%) of different object reference types in the error description sub-task without the tool and with the tool.	120
6.3	Relative frequencies (%) of different object reference types in the error fixing sub-task without the tool and with the tool.	122

1 Introduction

1.1 BACKGROUND

1.1.1 What are spreadsheet systems?

Spreadsheet systems are one of the most widely used applications used for trivial as well as non-trivial applications in private and public enterprises (Panko, 2000; Ballinger, Biddle, & Noble, 2003; Wilson et al., 2003). They are used for a variety of important tasks such as business decision making, mathematical modelling, scientific computations, tabular and graphical data representation, data analysis among others (Chambers & Scaffidi, 2010).

Spreadsheet systems allow computations to be defined by cells and their formulae. A cell's value is defined solely by the formula explicitly given to it by the user (Burnett et al., 2001). A cell value is recalculated automatically whenever a value on which it depends (a reference) changes thus providing immediate feedback. Spreadsheet systems also provide for copying of contiguous regions of cells from one physical area to another. References between the cells may be either absolute or relative in either their horizontal or vertical index. All copies of an absolute reference will refer to the same row, column or cell whereas a relative reference refers to a cell with a given offset from the current cell.

A spreadsheet program (hereafter referred to synonymously as spreadsheet) is usually perceived only as a two-dimensional grid of cells populated mainly with numerical values although every spreadsheet has a formula view as well as an underlying data-flow graph (Igarashi, Mackinlay, Chang, & Zellweger, 1998). An illustration of different views of a spreadsheet by Igarashi et al. (1998) is presented in Figure 1.1. A data-flow graph represents the network-structure of cell dependencies expressed by the references in the individual formulae. However, the data-flow graph is normally "hidden" from the spreadsheet developer. It is therefore not surprising

that most spreadsheet developers superficially view a spreadsheet as a word processor for numbers and not necessarily as a complex data-flow graph that spreadsheets really are (Clermont, 2005).

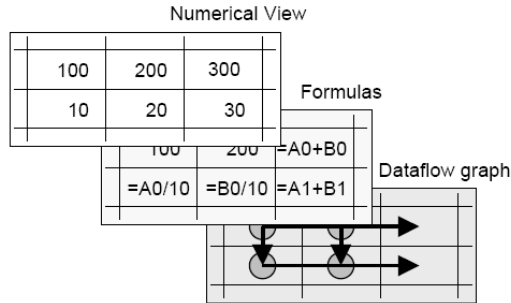


Figure 1.1: An illustration of different views of a spreadsheet.

Like the numerical view of a spreadsheet, the formula view of a spreadsheet has also some disadvantages. It is possible to see either all the formulae or all the values but not both at the same time. For a single cell, it is possible to see both the formula and the cell value at the same time but this does not give much information about the overall structure of the spreadsheet. In some cases, this locality to a single cell may help by narrowing the point of focus instead of dealing with the spreadsheet as a whole, but it is also difficult to get sense of the general structure of the whole spreadsheet (B. A. Nardi, 1993; Hendry & Green, 1994). As a result, it is difficult to identify where data comes from and where it goes unless one makes a detailed examination of the cell dependencies.

The spreadsheet paradigm also differs from the procedural programming paradigm in several ways:

- Spreadsheet programs are modelless in the sense that they do not require the spreadsheet developer to separately code, compile, link, and execute the spreadsheet program as is the case with procedural programs (Ruthruff et al., 2005).
- Spreadsheet programs provide immediate feedback to the spreadsheet developer. For example, when a formula for a

particular cell changes, the results are immediately reflected (Ruthruff et al., 2005).

- The structure of a spreadsheet program is usually represented in a two-dimensional tabular layout while the code for procedural programs is represented in a linear fashion (Ayalew, 2007).
- From the point of view of a spreadsheet developer, a spreadsheet program does not have clear separation between input, computational code and output. This is not the case with procedural programs (Ayalew, 2007).

1.1.2 Importance of spreadsheets

Although some spreadsheets are simple throw-away scratch-pad calculations, many spreadsheets have been quite useful for business as well as personal endeavours (Wilson et al., 2003). There are some large periodically used spreadsheets that are submitted to regular update-cycles like any conventionally evolving application software (Clermont, Hanin, & Mittermeier, 2002).

Panko (2000) conducted a study in which it was observed that 46 percent of non-trivial spreadsheets examined were rated as important or very important to the surveyed organization. Panko (2000) also noted that another study found that information generated from spreadsheets is also used in high-level decision making offices in business enterprises. This shows how critical non-trivial spreadsheets can be, to the running of a business enterprise.

Spreadsheets have also been used in science and engineering disciplines such as physics and chemistry, just to mention a few, because spreadsheets are assumed to be more usable than procedural programs (Clermont et al., 2002). Another reason for spreadsheet usage in science and engineering is that spreadsheets already incorporate a way of displaying graphs and this can be very useful in displaying results of scientific experiments (Clermont et al., 2002).

The preceding examples demonstrate that spreadsheet programming can not be regarded as a trivial subject.

1.1.3 Deficiencies in spreadsheets

Despite their popularity, most spreadsheets have deficiencies. For example, most spreadsheets are created by people who are not professional programmers and as such, they contain errors which the developers themselves may not easily notice (Galletta, Hartzel, Johnson, Joseph, & Rustagi, 1996; Panko, 1998, 2000; Powell, Baker, & Lawson, 2009).

Studies have also shown that a high proportion of observed errors in spreadsheets are concerned with the construction and use of formulae (Chadwick, Knight, & Rajalingham, 2001). Unfortunately, most spreadsheet errors are not trivial considering that key decisions, for example in business firms, are based on information extracted from spreadsheets (Galletta et al., 1996; Panko, 1998, 2000; Powell et al., 2009). Therefore, it is important to help spreadsheet developers expose these errors or even prevent them from occurring as errors in spreadsheets may lead to flawed decision making.

Furthermore, non-trivial spreadsheets may need to be modified by other people other than the spreadsheet developer. Moreover, changes to the structure of the spreadsheet may be necessary since spreadsheets may need to be maintained just as any conventionally evolving application software (Clermont et al., 2002). However, for one to make meaningful changes to the structure of a spreadsheet, it is necessary to understand or comprehend the spreadsheet first (Davis, 1996).

Spreadsheets normally come in the two-dimensional tabular arrangement of numeric values with some accompanying explanatory text. Usually this does not suffice for a third party to clearly comprehend and understand what the spreadsheet is all about.

Thus, problems of understandability or comprehensibility are one key deficiency in spreadsheets which unfortunately also compound the problem of errors in spreadsheets (Davis, 1996; Clermont et al., 2002).

1.1.4 Impact of deficiencies in spreadsheets

Errors in spreadsheet programs are non-trivial and costly (Galletta et al., 1996; Panko, 1998, 2000; Powell et al., 2009). Despite this observation, there has not been quantitative data on the impact of spreadsheet errors. However, the European Spreadsheet Risks Interest Group (EuSpRIG) publishes on its web page¹, verified stories on how errors in spreadsheets have affected public as well as private enterprises.

For example, it is documented on the website that in 2004, some city officials, in one of the cities in the United States, miscalculated the amount of sales taxes generated at one of the city's parks during the first couple of months of its operation. The mistake inflated the figures by tens of thousands of dollars, which in turn meant the total sales estimates were overblown by millions of dollars. The mistake was attributed to an error in a spreadsheet formula which amplified a subtotal amount.

It is also documented that mis-stated earnings of a company led to the stock price of an online retailer to fall by 25 percent in a day and the Chief Executive Officer had to resign. Again a spreadsheet error was to blame. A single erroneous numerical input in a spreadsheet was the cause of the mis-statement. These are just some of the stories that underscore that spreadsheet errors are non-trivial and costly.

1.1.5 Classification of errors in spreadsheets

Data from field studies and laboratory experiments indicate that errors in spreadsheets are indispensable. Panko (2000) has tabulated data indicating error rates in spreadsheets as produced by researchers in various field audits and laboratory experiments. The most important result of these studies is that spreadsheet error rates are huge enough to tell us that most non-trivial spreadsheets will contain errors.

¹<http://www.eusprig.org/horror-stories.htm>

Several classification schemes have been identified to categorize spreadsheet errors depending on the context in which a researcher is doing the analysis (Ayalew, Clermont, & Mittermeier, 2000). Panko (2000) identified three categories of spreadsheet errors namely: mechanical errors, logical errors and omission errors. Mechanical errors are simple slips such as mistyping a number or pointing to a wrong cell when entering a formula. Logical errors are defined as errors that occur when a spreadsheet developer has a wrong algorithm for a particular formula cell. On the other hand, omission errors are defined as errors that occur when a spreadsheet developer does not have complete understanding of the problem at hand and therefore produces an incomplete spreadsheet model of the problem. Hence omission errors are introduced due to faulty reasoning.

Another general classification scheme used by Panko (2000), categorizes spreadsheet errors as quantitative errors and qualitative errors. A quantitative error is defined as an error that produces an incorrect value in at least one bottom-line variable in a spreadsheet. On the other hand, qualitative errors emanate from factors such as poor spreadsheet design which may later cause problems in data entry or even lead to incorrect data modifications and hence generate quantitative errors. This scheme further categorizes quantitative errors into mechanical, omission and logical errors which have already been defined in the preceding paragraph.

Raffensperger (2008) classified spreadsheet errors into five categories: wrong input errors, accidental logic errors, wrong thinking errors, accidentally overwritten formula errors and software errors. Wrong input errors are caused when one types in a wrong value as input and also when spreadsheet data goes stale due to a spreadsheet containing time-sensitive data going out of date. Accidental logic errors are caused when a formula points to a wrong cell, column, row or blank cell. Errors of this type can also be caused by a numerical formula pointing to a label when it should point to a number and also when a formula is wrong such that it does not reflect what is true in the problem domain. Wrong thinking errors

are caused when a whole spreadsheet is correct technically but it fails to solve the correct problem due to limited understanding of the whole problem being solved by the spreadsheet developer. Accidentally overwritten formula errors are caused when a formula is overwritten with a constant. Software errors, on the other hand, are caused by bugs inherent in the spreadsheet system in use. For example, a formula failing to update after inserting a row at the bottom of a block.

There is another spreadsheet error classification scheme proposed by Ayalew et al. (2000). Unlike the other classification schemes given above, Ayalew et al. (2000) do not categorize spreadsheet errors by their cause, but rather by the spreadsheet concept the errors seem to be associated with. Thus, they have three categories of errors namely: physical area related errors, logical area related errors and general errors.

Physical area related errors are defined as those errors that normally deal with missing values in a physical area or values of the wrong type somewhere in the physical area. This kind of errors leads to several side-effects such as impacting on the results if new values are added to the area. According to this classification scheme, physical area related errors include what are termed as “reference to a blank cell/reference to a cell with value of wrong type” errors, “incorrect physical area specification” errors, “accidental deletion/addition of a cell within a physical area” errors and “physical area mix up” errors. On the other hand, a logical area is defined as an area that represents some kind of cohesion between cells. It usually originates from copying from the same source multiple times. Examples of logical area errors include overwriting a formula with a constant value and having a formula copy mis-reference. General errors have been defined as those errors that are not explicitly associated with a physical or logical area and are usually made during formula definition. An error might occur due to typographical errors or inability to formulate the necessary mathematical expression for a formula. An error might also occur due to incorrect use of formats which might affect the way a value

is displayed.

1.1.6 Spreadsheets in the context of the end-user programming paradigm

Computer end-users may be defined as people for whom conventional computer programming is not their main job although they use computers as part of their daily lives (Blackwell, 2002). However, it is now common place to see computer end-users (hereafter referred to as end-users) being involved in some form of “programming”. End-users are being involved in “programming” applications such as spreadsheets, databases, animations, web applications, simulations, just to mention but a few. Although end-users are not professional programmers they might be experts in their professional domains. Some of these end users are business professionals, educators, scientists, engineers and many more belong to other professions.

End-users may be motivated to do some “programming” because they want to use a computer to accomplish a particular goal. For example, a teacher may create a spreadsheet for recording student grades for a particular course. End-users may also do some “programming” because this might be an efficient way of solving a problem in comparison to manually solving the problem. For example, a mathematician may write some program code using a mathematical software application to find a solution to a complex differential equation. In all these cases, their main goal would be accomplishing a task at hand rather than producing high-quality, dependable program code (Ko, 2007). Pre-packaged software applications may not be suitable in these situations because these software applications cannot do every task required by an individual and worse still, they cannot be customized to every individual’s needs (Myers, Ko, & Burnett, 2006). This need has led to the birth of the end-user programming paradigm.

The rising growth in the popularity of the end-user programming paradigm can be attributed to the tools that have been developed to empower this kind of computer users. For example, the

development of the spreadsheet paradigm has led to many users developing their own spreadsheets, hence doing some “programming”. An end-user programming environment provides the tools for an end-user to accomplish a task at hand. Examples of end-user programming environments include spreadsheet systems, web authoring applications and animation environments.

Statistically, it was estimated that, in the year 2005, there were 55 million end-user programmers in the United States alone. This was about 20 times greater than the estimated number of professional programmers (Wilson et al., 2003; T. Y. Chen, Kuo, & Zhou, 2005; Scaffidi, Shaw, & Myers, 2005). These estimates clearly indicate that a sizeable amount of software produced in the whole world is developed by non-professional programmers. These end-user programmers write programs not as their primary job function but rather to support their quest for achieving their main goal such as accounting, doing office work, developing a web page, etc. (Myers et al., 2006). And indeed, although the art of creating a spreadsheet is a programming activity, many spreadsheet authors are experts in their own occupation but not in programming tasks hence they are end-user programmers (Sajaniemi & Pekkanen, 1988; Green, 1990).

1.1.7 Challenges in end-user programming

Despite the huge popularity in end-user programming, programs developed by end-users are very prone to errors (Galletta et al., 1996; Panko, 1998, 2000; Powell et al., 2009). Programs developed by end-users are not developed according to software engineering principles as is the case with software developed by professional software developers. Many end-user developers would not want to get involved in the nitty-gritties of coding in a particular programming language, let alone try to learn the formal syntax and semantics of a particular programming language. In fact, learning programming language syntax has been identified as one of the significant learning barriers in end-user programming environments (Ko, 2007). Other learning barriers in end-user programming environments include (Ko, 2007):

- Design barriers: the end-user programmer might *not know* what he/she wants the computer to do in order to solve a problem.
- Selection barriers: the end-user programmer might know what he/she wants the computer to do but does not know how to *choose* an appropriate tool for the task.
- Coordination barriers: the end-user programmer might know the appropriate tools for a particular task but he/she does not know how to make the tools *work together* in order to solve the problem at hand.
- Use barriers: the end-user programmer might know what tools to use for a particular task but does *not know how to use* those tools.
- Understanding barriers: the end-user programmer might think that he/she knows how to use a particular tool but unfortunately the *tool does not do what he/she expects*.
- Information barriers: the end-user programmer might think that they know why a tool behaved in an unexpected or problematic manner but they might *not have knowledge* to check the problem.

Another major challenge in end-user programming is the reality that end-user needs vary so widely such that one cannot come up with general design tools and languages that can fit every end-user programmer's needs (Ko, 2007). It is also a major challenge to make users understand the importance of the programs they develop (Ko, 2007). This is particularly true for non-trivial programs that have long life spans such that the programs might need long-term maintenance. A case in point are some spreadsheets that are not simple throw-away calculations but are continuously evolving as part of a business enterprise reporting function. The question would therefore be on how to develop tools that can capture the evolving program's history and design (Ko, 2007).

In trying to address some of the challenges outlined above, some researchers have advocated that much research on end-user programming should focus on development environments which can help end-users achieve their goals through the use of metaphors such as forms and spreadsheets (Segal, 2005). Coupled with the statistics on the number of end user programmers, it is easy to see the need for more research in end-user programming.

1.1.8 From end-user programming to end-user software engineering

One research direction in end-user programming has been the concept of “end-user software engineering” (Burnett, Cook, & Rothermel, 2004). End-user software engineering attempts to answer the research question “Is it possible to bring the benefits of rigorous software engineering methodologies to end users in order to address the problem of lack of quality in software produced by end-user programmers?” (Myers, Burnett, & Rosson, 2005).

End-user software engineering does not propose to transform end users into software engineers (Burnett, 2009). Rather, it advocates for the creation of end-user programming environments that employ new, incremental, feedback devices supported by analysis and inferential reasoning to help the end-user reason about the dependability of their software as they work with it, in a manner that respects the end-user’s problem-solving directions to an extent unprecedented in existing software development environments (Burnett, 2009).

Thus, end-user software engineering can be formally defined as end-user programming that involves systematic and disciplined activities that address software quality issues such as reliability, efficiency, usability, etc. (Ko et al., 2011). End-user software engineering can therefore be viewed as an extension of end-user programming in the sense that end-user programming is the “create” part of end-user software development, and end-user software engineering adds consideration of software quality issues to both the “create” and the “beyond create” parts of end-user software devel-

opment (Burnett, 2009).

It is important, however, to note that a basic premise in end-user software engineering is that it can only succeed to the extent that it acknowledges that the end-user probably has little expertise or interest in software engineering (Burnett, 2009). Thus, the challenge of end-user software engineering research is to find ways to incorporate software engineering activities into users' existing workflow, without requiring people to substantially change the nature of their work or their priorities. For example, rather than expecting spreadsheet users to incorporate a testing phase into their programming efforts, spreadsheet tools can simplify the tracking of successful and failing inputs incrementally, providing feedback about software quality as the user edits the spreadsheet program (Ko et al., 2011). To this end, several end-user software engineering tools and techniques have been developed, even in the spreadsheet context.

1.1.9 Proposed end-user software engineering tools in the spreadsheet context

As the adage says "prevention is better than cure", some end-user software engineering tools have focussed specifically on preventing spreadsheet users from making errors in their spreadsheets. Erwig, Robin, Irene, and Steve (2005) developed a system called Gencil in which spreadsheet templates using the Visual Template Specification Language (ViTSL) are used to generate spreadsheets which are free from reference, range or type errors. With this technique, spreadsheet templates are created and verified by domain experts and later on can be used by less experienced users to generate spreadsheets that always conform to the template. This concept was extended to include the automatic generation of spreadsheet templates from object-oriented specifications that have been specified using Unified Modeling Language (UML) diagrams (Engels & Erwig, 2005). This model driven spreadsheet environment was called ClassSheets whereby the business logic of a spreadsheet is captured through templates (Engels & Erwig, 2005).

Belo et al. (2013) introduced a query language based on the

Structured Query Language (SQL) where users can easily construct queries right in their spreadsheet environment, without the need for complicated configurations, or extra programs other than a simple add-on. Their approach, called QuerySheet, builds upon the concept of ClassSheet models whereby queries refer to entities in ClassSheet models, instead of the actual data, and thus allowing the user not to have to worry about the arrangement of the spreadsheet's data, but only what information is present (Cunha, Erwig, & Saraiva, 2010; Belo et al., 2013). Model driven approaches have the advantage of guaranteeing spreadsheets which are free from certain types of errors since spreadsheet users can only work on spreadsheets that conform to predefined specifications. Nevertheless, we put forward that model-driven approaches may be problematic to some spreadsheet developers since model-driven approaches provide an environment that is different from the traditional spreadsheet environment which provides for spreadsheets to be developed in an ad-hoc, interactive and incremental manner.

Other end-user software engineering tools have focussed on automatically detecting errors in spreadsheets for the user. Ayalew and Mittermeier (2003) developed a spreadsheet debugging technique based on "interval testing" and slicing. In this technique, each formula cell has a user-specified value interval and a system-generated value interval. When the user-specified interval and the system-generated interval for a cell do not agree with the actual spreadsheet computation, the cell is marked as displaying a symptom of a fault. A fault tracing strategy is then used to identify the most influential faulty cell from the cells perceived by the system to contain faults. This is based on the number of precedents and dependents of the influential faulty cells. Automatic visualization of potential faulty cells in spreadsheets is advantageous to spreadsheet users as it helps the user to quickly spot potential faulty cells in spreadsheets. However, requiring users to be specifying value intervals may be problematic to some spreadsheet users as user intervention requires extra effort from spreadsheet users.

Hofer, Ribeira, Wotawa, Abreu, and Getzner (2013) adapted and

applied to spreadsheets, debugging techniques designed for more traditional procedural or object-oriented programming languages by using a more formal approach with similarity coefficients to calculate fault probabilities of cells in a spreadsheet. The techniques adapted are spectrum-based fault localization, spectrum-enhanced dynamic slicing, and constraint-based debugging. Other researchers like Jannach and Schmitz (2014) have developed automatic fault localization tools for spreadsheets by translating a given spreadsheet into a constraint-based representation, such that additional inferences about possible reasons for an unexpected value in some of the cells can be made. Automatic fault localization tools have the advantage of helping spreadsheet developers to quickly spot potential errors in spreadsheets. However, they do not guarantee to catch all errors in spreadsheets. Moreover, the spreadsheet developer sometimes has to provide some information in advance e.g. the expected outcomes or which cells produce a correct output and which cells are erroneous (Jannach, Schmitz, Hofer, & Wotawa, 2014).

Abraham and Erwig (2004) developed an automated reasoning system for spreadsheets called UCheck. UCheck infers header unit information for cells based on labels in a spreadsheet. Based on the header unit information, the system identifies cells in the spreadsheet that contain erroneous formulae. Abraham and Erwig (2005b) also extended the UCheck system to produce a system known as UFix in order to improve on the way error messages are reported to users hence improving the spreadsheet debugging process. They also developed a type system and a type inference algorithm for spreadsheets which can be used in identifying some kind of errors in spreadsheets (Abraham & Erwig, 2004).

Abraham and Erwig (2005a) also developed a semi-automatic spreadsheet debugger known as GoalDebug based on a technique known as "goal-directed debugging". GoalDebug allows users to mark cells with incorrect outputs and specify the expected output. The GoalDebug system then generates a list of change suggestions, any one of which when applied would result in the expected output

being computed in the marked cell. The generated change suggestions are ranked based on a set of heuristics before being presented to the user. The generated change suggestions can be automatically applied and hence eliminating errors that can be introduced by end users through editing of cell formulae. Techniques such as GoalDebug can be of help to spreadsheet users as they can quickly spot potential errors in spreadsheets. However, expecting users to provide information such as expected output beforehand could be tedious on the part of the user.

Some end-user software engineering tools have focussed on helping spreadsheet users to inherently test their spreadsheets.

Rothermel, Burnett, Li, Dupuis, and Sheretov (2001) developed a spreadsheet testing methodology which they termed "What You See Is What You Test" (WYSIWYT) to help users test spreadsheets. The methodology uses data-flow adequacy and coverage criteria to give the user feedback on how well tested a spreadsheet is. The WYSIWYT testing methodology has been integrated with another spreadsheet testing technique known as the "Help Me Test" (HMT) (Fisher, Cao, Rothermel, Cook, & Burnett, 2002) technique into the Forms/3 (Burnett et al., 2001) spreadsheet language.

The HMT technique automatically generates test cases for the user as he/she actively works on the spreadsheet. Forms/3 is a form-based research spreadsheet language developed at the Oregon State University that also allows users to define assertions on expected cell values (Burnett et al., 2003). To promote the usage of assertions by end-user programmers, Wilson et al. (2003) devised a curiosity centred approach to eliciting assertions from end-users through a "surprise, explain, reward" strategy. Randolph, Morris, and Lee (2002) developed a spreadsheet verification tool based on the WYSIWYT methodology. Their main emphasis was to use the WYSIWYT methodology algorithms in implementing a spreadsheet independent tool. They placed much emphasis on issues of portability and the automatic generation of test cases.

Ruthruff et al. (2005) transferred the concept of program slicing to spreadsheets to come up with a visual, interactive approach

to fault localization within the WYSIWYT methodology context. Their technique also required a user to specify in advance information about correct and incorrect cell values and considered those cells that theoretically contribute to an erroneous cell value to be possibly faulty. Incorporating testing as an inherent part of the spreadsheet process as implemented in testing methodologies such as WYSIWYT methodology can be advantageous to the user as it is not intrusive to the spreadsheet developer hence could be easily adopted by spreadsheet users. This is because spreadsheet developers being non-professional programmers many not be interested in embarking on explicitly testing a spreadsheet.

Various spreadsheet visualization tools have also been proposed for different purposes such as spreadsheet comprehension, debugging, documentation, etc. all in the quest to improve software quality in spreadsheets. Generally, visualizing a program helps the user to check the correctness of a program by visualizing its behaviour (Ko et al., 2011). Most of spreadsheet visualization tools are based on the data-flow (data dependency) graph behind the spreadsheets (Sajaniemi, 2000).

Sajaniemi (2000) developed the S2 and S3 spreadsheet visualization tools in which logical areas or semantic units in a spreadsheet are highlighted and data-flow between logical areas is indicated through arrows. This was an improvement to the arrow tool by Davis (1996).

Y. Chen and Chan (2000b) developed a tool that enabled all precedents and dependents of a group of cells to be shown at once, instead of just displaying arrows between individual cells as done, for example, in Microsoft Excel. Ayalew et al. (2000) proposed a graphical spreadsheet visualization model that is not only based on a data-flow graph but also on visualizing logical and physical areas in spreadsheets. Clermont et al. (2002) developed a spreadsheet visualization toolkit that partitions a spreadsheet into logical areas known as equivalence classes. The equivalence classes are mainly based on structural similarity of formulae. With large spreadsheets (e.g. having more than 5000 used cells), the number of equivalence

classes becomes too large and hence they devised a further abstraction mechanism called semantic classes. Semantic classes are represented as nodes in a generated graph and data-flow between cells in different semantic classes is represented by directed edges.

Ballinger et al. (2003) developed a spreadsheet visualization tool that would first statically extract artifacts from spreadsheets and then convert this information into visualizations such as spreadsheet data-flow diagrams. Kankuzi and Ayalew (2008a, 2008b) also developed a spreadsheet visualization tool in which semantically related cells in a spreadsheet are automatically detected and highlighted by using the Markov Clustering (MCL) algorithm to aid in spreadsheet comprehension.

Hermans, Pinzger, and van Deursen (2011) also developed a spreadsheet visualization tool based on data dependencies whereby the user can inspect the data flows within a spreadsheet on different levels of detail – from a global view, different worksheets of a spreadsheet and their dependencies are shown; on the lowest level, the formula view, dependencies between individual cells are displayed while on an intermediate level, the worksheet view, all data blocks within a worksheet, as well as the dependencies between them are shown. Hermans, Pinzger, and van Deursen (2012) also extended their tool to automatically detect potential sources of errors in the general design of spreadsheets containing many worksheets through what they called “inter-worksheet smells”.

As the saying goes that “a picture is worth a thousand words”, visualizing spreadsheets provide glimpses on different aspects of a spreadsheet and hence may aid in spreadsheet comprehension and debugging. However, we put forward that spreadsheet visualizations need to be developed in such a way that they are easy to use and empirically proven to be useful to spreadsheet users.

1.2 STATEMENT OF THE PROBLEM

As alluded to above, despite their widespread popularity, spreadsheets exhibit some deficiencies. Most spreadsheets contain non-

trivial errors which the developers themselves may not easily notice (Galletta et al., 1996; Panko, 1998, 2000; Powell et al., 2009). This problem of errors in spreadsheets is, however, compounded by spreadsheets being generally difficult to understand and comprehend, despite the simplicity in creating them and their intuitive interface (Davis, 1996; Clermont et al., 2002). Studies have also shown that a high proportion of observed errors in spreadsheets are concerned with the construction and use of formulae (Chadwick et al., 2001).

Many tools and techniques have been developed to try to correct these deficiencies but the problems still persist. In this research work, we tried another approach, i.e., study whether it is possible to devise spreadsheet tools based on spreadsheet developers' mental models. Realizing that the term "mental model" has been defined differently by different researchers, in our context, we borrow the definition by Doyle and Ford (1998) which states that a mental model is "a mental image of the world around us that we carry in our heads depicting only selected concepts and relationships that represent real systems." Accordingly, a user's mental model of a system reflects the user's understanding of what the system contains, how it works and why it works that way (Carroll, Olson, & Anderson, 1987). Thus, for example, a spreadsheet developer's mental model for a spreadsheet does not carry all possible information, but just those aspects of the spreadsheet that the developer finds appropriate for the task he or she has. It is important for features of a system to match with the corresponding mental models of its users as otherwise there will be mental conflicts resulting in errors and other sub-optimal behaviour (B. A. Nardi & Zарmer, 1993).

1.3 RESEARCH OBJECTIVES AND QUESTIONS

To attack the problem at hand, we chose to study spreadsheet authors' mental models as they are doing various spreadsheet tasks whether when working with a spreadsheet created originally by

themselves or by someone else. The term *spreadsheet author* can be used synonymously with *spreadsheet developer*. However, in this research work, we opt for the term spreadsheet author to emphasize distinction from a professional software developer and also to emphasize that spreadsheet authors do not just work on spreadsheets created by others but also create non-trivial spreadsheets themselves and use these in their daily work and hence they are particularly familiar with writing and editing spreadsheet formulae.

It is a common assertion that humans have mental models of the systems they interact with and it is difficult to explain many aspects of human behaviour without resorting to a construct such as mental models (Rouse & Morris, 1986). We therefore argue that it is important to first of all understand what types of mental models do spreadsheet authors possess when they are doing different spreadsheet tasks in order to better understand why the spreadsheet process is so error-prone and to be able to devise new tools that better correspond to their mental models. Therefore, in this research work, we applied the theory of mental models to try to come up with a solution to the spreadsheet error and comprehension problem.

Our research work had three specific objectives:

- (i) We wanted to investigate and characterize mental models of spreadsheet authors as they are doing various spreadsheet tasks, and particularly, when they are explaining and debugging their spreadsheets.
- (ii) We wanted to develop a spreadsheet visualization tool that demonstrates that it is possible to devise spreadsheet authoring and debugging tools that are easy to use, efficient and correspond to spreadsheet authors' mental models of spreadsheets.
- (iii) We wanted to put forward a case if there is need to shift from the traditional spreadsheet paradigm to another paradigm that can better reflect spreadsheet authors' mental models.

For each objective, we had specific research questions that had to be answered in order to achieve the set goal. In investigating and characterizing mental models of spreadsheet authors we had the following research questions:

- **RQ1** – *Do spreadsheet authors have several mental models when they are doing various spreadsheet tasks, and particularly, when they are explaining and debugging their spreadsheets?*
- **RQ2** – *What are the roles of mental models in spreadsheet authors when they are doing various spreadsheet tasks, and particularly, when they are explaining and debugging their spreadsheets?*
- **RQ3** – *What are the implications of mental models in spreadsheet authors on spreadsheet authoring and debugging tools?*

In a quest to develop a spreadsheet visualization tool that demonstrates that it is possible to devise spreadsheet authoring and debugging tools that are easy to use, that are efficient and that correspond to spreadsheet user's mental models of spreadsheets, we had the following research questions:

- **RQ4** – *Is it possible to develop a spreadsheet understanding and debugging tool that relieves users from spreadsheet details and lets them utilize more of their mental model of the application domain?*
- **RQ5** – *Can the tool be learned easily by users?*
- **RQ6** – *Can the tool help users in debugging their spreadsheets?*
- **RQ7** – *Can spreadsheet authors be satisfied with the tool after using it?*

In a quest to put forward a case if there is need to shift from the traditional spreadsheet paradigm to another paradigm that can better reflect spreadsheet authors' mental models, we had the following research questions:

- **RQ8** – *Is it possible to construct a tool that is based on a different paradigm rather than the current traditional spreadsheet paradigm?*

- **RQ9** – *What are the effects of such a tool on spreadsheet authors' mental models?*

Research questions **RQ1**, **RQ2** and **RQ3** are answered in Chapter 3; research questions **RQ4**, **RQ5**, **RQ6** and **RQ7** are answered through the work presented in Chapter 4 and Chapter 5; while research questions **RQ8** and **RQ9** are answered in Chapter 6.

1.4 RESEARCH METHODOLOGY

Constructive research is the overall approach that has been used in this work. Constructive research is a research methodology that aims at producing novel solutions to practically and theoretically relevant problems and it is widely used in software engineering and computer science although it has its origins in management and social sciences (Kasanen & Lukka, 1993; Iivari, 2007). This methodology is about *constructing* an artifact that solves a domain problem in order to create knowledge about how the problem can be solved, and if previous solutions exist, how the solution is new or better than previous ones (Kasanen & Lukka, 1993; Iivari, 2007).

The essence of constructive research are *artifacts* such as a programming language, a compiler, a software development method, an algorithm, etc. In the use of artifacts, there are domain problems discovered and knowledge is created as a response to the particular domain problem of the particular artifact (Kasanen & Lukka, 1993; Labro & Tuomela, 2003; Piirainen & Gonzalez, 2013).

Constructive research usually has got the following steps (Kasanen & Lukka, 1993; Labro & Tuomela, 2003; Piirainen & Gonzalez, 2013):

- (i) Finding a practically relevant problem.
- (ii) Obtaining an understanding of the topic and the problem.
- (iii) Innovating, i.e., constructing a solution idea based on a solid theoretical justification.

- (iv) Demonstrating that the solution works.
- (v) Showing theoretical connections and research contribution.
- (vi) Examining the scope of applicability and generalizability.

In practice, though, the steps do not follow each other in a simple sequence as the process is both iterative and sometimes recursive (Kasanen & Lukka, 1993; Piirainen & Gonzalez, 2013).

In *looking for problems*, one can find a problem through literature search, colleagues, their own experience, etc. and most importantly the problem has to be practically relevant.

In *understanding the topic and the problem*, one can be practical by doing *empirical work*, for example, through a user study or observation (participation). In our case, we mainly used “user studies” in the empirical work. One can also be theoretical in understanding the problem by scanning through “relevant” literature to get a big picture of existing knowledge.

Innovating is about *constructing* a solution that solves the identified problem with a clearly defined theoretical justification. In our case, we develop spreadsheet visualization tools that demonstrate a proposed solution to the problem at hand.

In *demonstrating that the solution works*, a testing or validation exercise is conducted. Normally, validation is perhaps the hardest part of constructive research and it has to be performed in industrial settings, whenever possible to ensure practical relevance. Analysis and experience are some of the techniques that are used in validating a solution construct and the *empirical model* (data, usually statistical, on practice or in controlled situation is collected) is one way that is employed in validating a solution construct. In this research work, we also employed user studies as an empirical technique in validating the solution constructs. In particular, we conducted user studies on the spreadsheet visualization tools we developed.

On showing *theoretical connections and research contribution* of the solution construct, novelty is crucial. However, there are many possibilities on the novelty of the solution construct: it could be an en-

tirely new idea though this is rare; it could be cross-domain knowledge sharing; it could be an improved idea or implementation or solution; or it could be an interesting research approach; or indeed something else.

On *examining the scope of applicability and generalizability*, it is worth noting that broad applicability is generally a good thing but not always good because there are many specific practical applications where narrow scope solutions are valuable.

Overall, constructive research is evaluated based on the construct or artifact which is checked for relevance (theoretical and practical), novelty and practical utility.

As alluded to above, this research work was conducted based on a constructive research methodology. And we mainly used “user studies” in the aspects that involved empirical work that we carried out within this constructive research methodology context. We employed “user studies” in understanding the topic and the problem and in validating the proposed solution constructs.

A user study is an experiment in which human subjects are under study and it conforms to the norms for empirical inquiry and the scientific method (McKenzie, 2007). Empirical inquiry is a means of gaining knowledge through direct or indirect observation or experience other than theory or pure logic (Goodwin, 2009). A scientific method consists of systematic observation, measurement and formulation, testing and modification of hypotheses (Glass, 1994; Cohen, Manion, & Morrison, 2013). Therefore a user study is based on observation which has to be done in a systematic way conforming to specific principles of reasoning.

In the user studies, we stick to the ideals and characteristics of a true experiment as described by Lazar, Feng, and Hochheiser (2010):

- A true experiment is based on at least one testable hypothesis or research question and aims to validate it.
- There are usually at least two conditions (a treatment condition and a control condition) or groups (a treatment group

and a control group).

- The dependent variables are normally measured through quantitative measurements.
- The results are analyzed through various statistical significance tests.
- A true experiment should be designed and conducted with the goal of removing the potential of biases.
- A true experiment should be replicable with different samples, at different times, in different locations and by different experimenters.

We use a within-subjects experimental design in the studies that we have conducted in this research work. A within-subjects experimental design, also known as repeated-measures design, is an experiment design in which all participants in a user study are exposed to every treatment or condition (Seltman, 2012). One of the greatest advantages of a within-subjects design is that it does not require a large pool of participants (Seltman, 2012). Generally, a similar experiment in a between-subjects design would require twice as many participants as a within-subjects design. A within-subjects design can also help reduce errors associated with individual differences (Seltman, 2012). In a between-subjects design where individuals are randomly assigned to a treatment condition, there is still a possibility that there may be fundamental differences between the groups that might impact the results. In a within-subjects design, individuals are exposed to all levels of a condition, and as a result, the results will not be distorted by individual differences because each participant serves as his or her own baseline.

However, there are also drawbacks of a within-subjects design (Seltman, 2012). A major drawback of using a within-subjects design is that the sheer act of having participants take part in one condition can impact performance or behavior on all other conditions, a problem known as carryover effects. Second, fatigue is another

potential drawback of using a within-subjects design. Participants may become exhausted, bored or simply disinterested after taking part in multiple treatments or tests. Lastly, performance on subsequent tests can also be impacted by practice effects. Taking part in different levels of the treatment condition or taking the measurement tests several times might help the participants become more skilled. This can skew the results and make it difficult to determine if any effect is due to the different levels of the treatment or simply a result of practice. We carried out our studies with these drawbacks in mind and tried as much as possible to counterbalance them.

The user studies we have been conducting in this research work were following typical steps in the life-cycle of an human-computer interaction experiment (Lazar et al., 2010):

- (i) Identifying a research problem to be investigated.
- (ii) Specifying the design of the study such as selection criteria of participants for the study, instruments to be used and type of experimental design to be followed.
- (iii) Running pilot studies before rolling out the user studies in order to test the design and study instruments.
- (iv) Recruiting of participants in a user study - recruitment was mainly through word of mouth. Participants in our user studies were mainly professional accountants who use spreadsheets in their daily work.
- (v) Running the actual data collection sessions - each participant was being visited at their place of work to avoid introducing ecological effects in the studies. In the first user study, participants would talk aloud while performing a given task and the sessions would be recorded therein (verbal protocols (Rouse & Morris, 1986)). In the second user study, participants were subjected to a usability (Nielsen, 1994) evaluation of the tool we have developed through efficiency tasks such as locating errors in a spreadsheet with and without a tool. In the third

user study, participants performed debugging tasks and written transcripts (Rouse & Morris, 1986) were collected from the participants.

- (vi) Analyzing the data - for the first user study, transcripts of recorded sessions were encoded and analyzed through content analysis techniques adapted from program summary analysis (Good, 1999). For the second user study, usability metrics (Nielsen, 1994) were numerically analyzed and content analysis techniques used in other aspects of the analysis. For the third user study, content analysis techniques were used to analyze the written transcripts. Statistical analysis of data was done using the R Statistical Package (The R Foundation, 2013).
- (vii) Reporting the results - the results are reported with various visual aids to aid in the interpretation of the results.

In summary, this research work was conducted based on a constructive research methodology with user studies playing a pivotal role in empirical aspects of the work. An illustration of the sequence of steps we followed in this research work within the constructive research methodology context is given in Figure 1.2. Inherently in the same figure (Figure 1.2), we also show the interrelationships between the research questions being answered in this research work.

1.4.1 Justification of methodology

Constructive research has several advantages over less direct empirical field research approaches (Lukka, 2000). With constructive research, the feasibility and actual working of a construct is tested in practice (real-world set-up) hence offering a pragmatic way of demonstrating truth by arguing that “what works is true”. Thus constructive research bridges theory with practice. This is unlike, for example, just designing a theoretical model in a laboratory environment. Constructive research also offers the possibility of gaining straight forward practical benefits through the artifacts that are constructed hence acting as an incentive to research participants to par-

Introduction

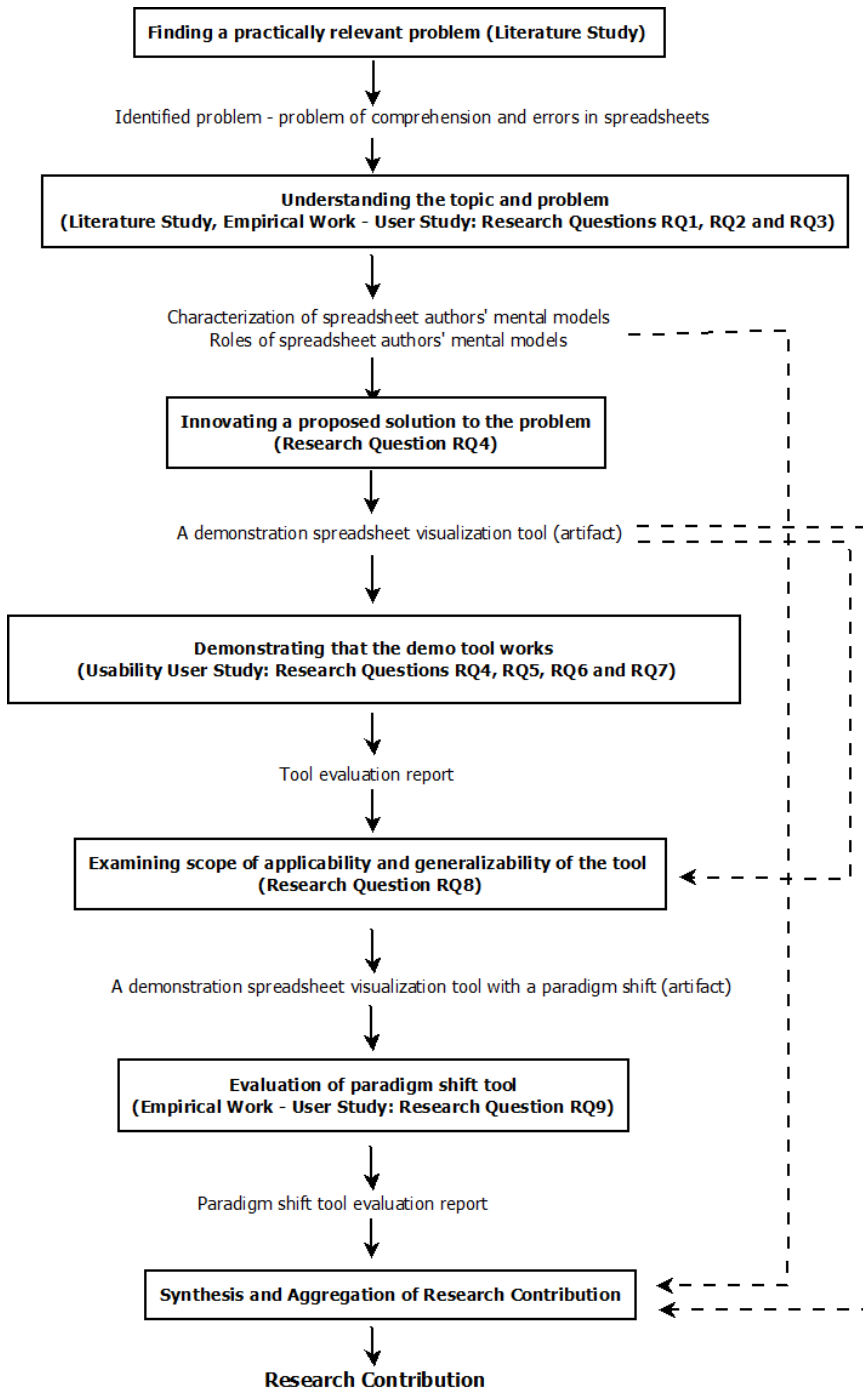


Figure 1.2: Steps we followed within the constructive research methodology context in this research work. Inherently are also interrelationships between the research questions.

ticipate in the studies as study participants expect that the constructive researcher helps to solve problems they face hence a win-win situation for both the researcher and research participants. This is unlike less direct empirical methods like questionnaire-based surveys where research participants may not have immediate direct benefit. It is for these benefits that we opted for constructive research methodology in this research work.

Despite the advantages of constructive research, there are also potential research risks of the constructive research approach (Lukka, 2000). For example, constructive research typically takes long time to conduct hence spanning several months to several years which may lead to researcher fatigue. A researcher following the constructive research approach also risks empty praising of findings of the research hence it is also important to maintain a sincere critical reflection of the findings revealed in the research process.

1.5 OVERVIEW OF THE REST OF THE DISSERTATION

The rest of the dissertation is organized as follows: Chapter 2 provides a review of various literature on the concept of mental models and how it is linked to programming in general and spreadsheets in particular. An empirical study of spreadsheet authors' mental models in explaining and debugging tasks that we conducted is presented in Chapter 3. The new spreadsheet visualization tool that we developed is presented in Chapter 4 while an evaluation of its usability and of its effects on mental models of spreadsheet authors is given in Chapter 5. We provide a case for the need for a paradigm shift in spreadsheets to a paradigm that better reflects spreadsheet authors' mental models in Chapter 6 while we conclude this dissertation in Chapter 7.

2 Literature Review

In this chapter, we review various literature on the concept of mental models and how it is linked to programming in general and spreadsheets in particular.

2.1 THEORY OF MENTAL MODELS

2.1.1 Nature of mental models

The term “mental model” was coined by Johnson-Laird (1983) although the history of the concept can be traced to the work by Kenneth Craik entitled “The Nature of Explanation” published in 1943 (Staggers & Norcio, 1993). Mental model theory challenged the prevalent notion that time that human reasoning depended on mental logic consisting of formal rules of inference similar to those in logical calculus (Johnson-Laird, 2010). Mental model theory has two main research threads: logical mental models and causal mental models (Markman & Gentner, 2001). *Logical mental models* are mental representations that people are thought to have when they perform logical reasoning tasks. On the other hand, *causal mental models* are mental representations that are used in reasoning about physical systems and mechanisms such as spatial representation, human-computer interaction, ecology and these mental representations are based on long-term domain knowledge or theories (Markman & Gentner, 2001). Causal mental models also differ from logical mental models in that logical mental models are created on the spot and involve only information currently active in working memory, whereas with causal mental models, even those currently active in working memory, are assumed to draw on long-term memory structures (Markman & Gentner, 2001). In this research work, we are focussing on causal mental models and not logical mental models and therefore the term “mental model” in this research work refers to “causal mental model”.

The study of mental models is important as it is a common assertion that humans have “mental models” of the systems they interact with and it is difficult to explain many aspects of human behaviour without resorting to a construct such as mental models (Rouse & Morris, 1986). The term “mental model” has, however, been defined differently by different researchers. Confusion has surrounded the term “mental model” because different researchers have sometimes referred to different owners of the models when talking about mental models (Carroll et al., 1987). For example, is the mental model being referred to that of an ordinary user of a system or that of a developer of a system? Confusion has also surrounded the term “mental model” because sometimes it is also not clear as to what the mental model represents (Carroll et al., 1987). For example, is the mental model representing a task or is it representing the architecture of a system? In our context, we borrow one definition given by Doyle and Ford (1998) where a mental model is defined as

“a mental image of the world around us that we carry in our heads depicting only selected concepts and relationships that represent real systems.”

Accordingly, a user’s mental model of a system reflects the user’s understanding of what the system contains, how it works and why it works that way (Carroll et al., 1987). It is therefore important for features of a system to match with the corresponding mental models of the user as otherwise there will be mental conflicts resulting in errors and other suboptimal behaviour (B. A. Nardi & Zamer, 1993). Systems whose features match with the corresponding user’s mental models of the same are easy to learn and use (Carroll et al., 1987).

Sometimes the term “mental model” is confused with the term “cognitive model”. However, these two terms are different. A mental model describes a user’s mental representation of a real system while a cognitive model describes the cognitive processes and information structures used to form the mental model (Storey, Frac-

chia, & Müller, 1999). Norman (1983) also distinguishes a “mental model” from a “conceptual model” by defining conceptual models as tools for the understanding or teaching of physical systems while mental models are what people really have in their heads and what guides their use of things. A user’s mental model of a system may be based on the conceptual model provided, but is probably not identical to it (Borgman, 1985). The term “mental model” is also not synonymous with the term “knowledge” in general (Rouse & Morris, 1986). On the other hand, the concept of “mental models” connotes special types of knowledge for describing a system purpose and form, explaining how a system works and its observed states, and prediction of future system states (Rouse & Morris, 1986). This research work focusses on mental models rather than cognitive models or conceptual models.

Mental models are thought to have some characteristics. Norman (1983) observed that mental models are incomplete since most people’s understanding of the systems they interact with is meagre and full of inconsistencies and sometimes this could lead one to have an inaccurate model of a system. Mental models are also thought to be unstable since many people forget about the details of the system they interacting with especially if they had not used the system after a period of time (Norman, 1983). In other words, mental models are dynamic and modifiable and are bound to change as one works with a system. Mental models are also thought to lack firm boundaries as mental models of similar systems and operations get confused with one another (Norman, 1983). Thus, a person can combine mental models of similar systems which sometimes leads to confusion.

Norman (1983) also put forward that mental models are “unscientific” in that people maintain “superstitious” beliefs about a system in the hope of saving mental and physical effort. Mental models are also thought to be “parsimonious” in that extra physical effort is often preferred over extra mental effort especially in relatively similar situations to avoid confusions (Norman, 1983). Mental models are also thought to be “runnable” or can be men-

tally simulated but human ability to “run” mental models is limited (Norman, 1983).

A person can also have simultaneous or parallel multiple mental models of a system which reflect the different levels of abstraction the person might have of a particular system when performing different tasks on the system (Carroll et al., 1987; Markman & Gentner, 2001). Thus, for example, one mental model could describe the technical aspects of a system and another the purpose of the system.

There are some issues which every researcher needs to consider when studying mental models. These include (Rouse & Morris, 1986):

- Accessibility of mental models - *To what extent is it possible to “capture” an individual’s mental models?* This is particularly constrained by human’s lack of ability to verbalize their mental models hence making it difficult for researchers to capture mental models. However, there are several methods that have been used to “capture” an individual’s mental models.
- Forms of representation of mental models - *What do mental models look like?* This issue concerns how mental models are encoded and how they evolve. This issue is important because accessibility and use of mental models is dependent on their form. For example, it is theorised that mental models are rather image-like or pictorial rather than symbolic in a list processing sense hence the difficulty in verbalizing mental models.
- Context of representation of mental models - *To what extent can mental models be general rather than totally context-dependent?* It is generally theorised that mental models are specific rather than general although this specificity is difficult in accounting for the richness in problem solving behaviour particularly in human ability to solve novel problems. However, this richness has been explained through the use of multiple men-

tal models, analogous problem solving and learning through metaphors.

- Nature of expertise - *How do mental models of experts differ from those of novices?* Evidence suggests that mental models of novices and experts are fundamentally different. It is theorised that experts tend to have conceptually abstract pattern oriented mental models while novices have representational mental models. However, this brings into question the use of verbalization techniques to capture mental models of experts since it is generally accepted that it is difficult to verbalize abstract “things”. It is also important to note that a shift from one being a novice to being an expert does not necessarily mean that all novice mental model aspects are discarded. In other words, mental models may include a bit of unnecessary “material” that humans might find no need to question or discard even though this might have negative effects in novel situations.
- Cue utilization - *How are mental models affected by the cues one employs either by choice or due to availability?* It is generally accepted that humans have that ability to extract from the environment cues necessary for predicting the future state of a system. Inappropriate cues could therefore lead to have wrong mental models on how something works. Cue utilization has also been found to be different between novices and experts. Experts seem not to be unduly influenced by superficial cues as compared to novices and therefore an important aspect of being an expert is to select more useful features of a problem or system.
- Instruction - *How can and should training affect an individual’s mental models?* In particular, *what mental models should a human have for a particular task or job and how should these mental models be imparted?* Normally it is assumed that an individual needs to be imparted with theoretical knowledge of a system

in order to form correct mental models of the system. However, there is little evidence that this results in better and more useful mental models. Therefore, a more holistic approach is needed by placing emphasis on the form of knowledge, guidance in the use of prior knowledge since all of these interact to affect the development and use of mental models.

2.1.2 Methods for “capturing” mental models

Human’s lack of ability to verbalize their mental models makes it difficult for researchers to capture mental models (Rouse & Morris, 1986). However, there are several methods that have been used to attempt to “capture” an individual’s mental models (Rouse & Morris, 1986) namely: empirical inquiry, empirical modeling, analytical modeling and direct inquiry.

Empirical inquiry involves the use of observational or experimental methods to “capture” the characteristics of mental models. Empirical studies provide evidence for effects of various independent variables on characteristics of mental models but only indirect insights into the form of mental models. This technique is widely used in experimental psychology.

Empirical modeling involves algorithmically identifying the relationship between what is observed in a person and subsequent actions. Techniques such as regression are used to identify input-output relationships and from these the structure and parameters of mental models can be inferred.

Analytical modeling involves using available theory and data to formulate assumptions about the form, structure and parameters of mental models for particular tasks. Based on these assumptions, human performance is calculated or computed analytically and compared to empirical data. The nature of some domains such as neural information processing dictate the use of analytical modeling.

Direct inquiry involves directly asking people about their mental models. This approach is a sharp contrast to the other indirect methods, i.e., empirical inquiry, empirical modeling and analytical

modeling. Introspection is used in direct inquiry. Introspection is the examination of one's conscious thoughts and feelings. There are several approaches to introspection. One of which is verbal protocol which is simply a transcript of a human "thinking aloud" as he or she performs a task. These verbal protocols serve as "data" for experiments. Various content analysis techniques can then be used to analyze the verbal protocols. The disadvantage with verbal protocols is that they provide information about what humans are thinking but not how they are thinking. Therefore, they should be used to test hypotheses on the "what" of thinking and not the "how" of thinking. Another approach to introspection is asking a person to write down what they are thinking (written transcript) instead of "thinking aloud". Another approach to direct inquiry is interviews and/or questionnaires. Interviews and/or questionnaires differ from verbal protocols in that with interviews or questionnaires the inquiry does not happen as the task is performed as is the case with verbal protocols and written transcripts.

The above stated methods can also be classified into two: empirical inquiry, empirical modeling and analytical modeling are inferential methods of "capturing" mental models while verbal protocols, written transcripts, interviews and questionnaires are verbalization methods (Rouse & Morris, 1986).

2.1.3 Limitations in the study of mental models

Studying mental models in human beings is also not without its limitations. First, there is always an effect of researcher subjectiveness in the study of mental models because in effect studying mental models of other people is tantamount to one or more humans developing models of other human's models of the external world (Rouse & Morris, 1986). Therefore researchers need to be aware of the biases they bring when they are studying the mental models of other people (Rouse & Morris, 1986). Second, the search for mental models will never ever eliminate uncertainty since it is difficult to actually "see" what is going on in someone's head (Rouse & Morris, 1986). And as such, it will always remain a

“blackbox” and therefore it is too simplistic to assume that mental models are static unitary entities that can be identified if appropriate methods are used (Rouse & Morris, 1986). It would rather be safe to assume that mental models are more likely to be dynamic entities that can have multiplicity of forms (Rouse & Morris, 1986).

2.1.4 Proposed research directions in the study of mental models

Carroll et al. (1987) recommended some research directions in the field of mental models. First, researchers should detail what a specific mental model would consist of and how a person would use it to predict a system’s behaviour. Second, researchers should investigate whether people have and use mental models of various kinds. Third, researchers should determine the behaviours that would demonstrate the models form and the operations used in it. Fourth, researchers should determine how people intermix different mental models in producing behaviour. Lastly, researchers should provide system designers with tools that can help them to develop system features that correspond to user’s mental models of a given system.

2.2 MENTAL MODELS OF PROGRAMMERS

There have been several studies that have focussed on mental models of programmers. It is important to understand mental models of programmers because programming is a highly cognitive activity that requires the programmer to develop abstract representations of a process in the form of logic structures and therefore having a well-developed and accurate mental model may affect the success of a programmer (Ramalingam, LaBelle, & Wiedenbeck, 2004).

Letovsky (1987) posited that a programmer has a mental model which encodes a programmer’s understanding of the target program. It is also quite agreeable among researchers that a programmer can have various simultaneous mental models of a program (Allen, 1997). These simultaneous mental models correspond to various levels of abstraction of a program by a programmer and

they are also dependent on the task at hand (Storey, 2005). It is also quite agreeable among researchers that programmers, whether expert or novice, usually have function related knowledge and program related knowledge about a program (Storey, 2005). This function related knowledge is also called the *domain model* since it captures the application or problem domain specific knowledge about the real-world problem being solved by a particular program. On the other hand, the program-related knowledge or *program model* captures the programmer's knowledge about program text and control flow.

Users might have different mental models of a system depending on the sophistication of the user (Carroll et al., 1987). Therefore, expert programmers do not necessarily have the same mental models as novice programmers. For example, an expert programmer might have very different understanding of a given computer program than a novice programmer.

2.2.1 Mental models of expert programmers

Brooks (1983) theorized that, in a program comprehension task, expert programmers understand a completed program in a top-down manner by reconstructing knowledge about the problem domain of the program (domain model) and mapping that to the actual program code itself (program model). In other words, in program comprehension, an expert programmer first constructs a domain model about the program before mapping it to a program model. The mapping goes through several intermediate domains, and comprehending a program involves reconstructing part or all of these mappings. Soloway and Ehrlich (1984) also found that expert programmers use a top-down approach in comprehending a program.

On the other hand, Pennington (1987) found that expert programmers comprehended procedural programs in a bottom-up manner such that a program model is formed first before a more abstract domain model. Bergantz and Hassell (1991) found that in a logic programming comprehension task, programmers also used a bottom-up approach by constructing both a program model based

on the detection of data structure relationships and a domain or real-world model based on the detection of function relationships.

Familiarity with a problem domain seems to affect the type of comprehension model an expert programmer can invoke. For example, expert programmers invoked the top-down comprehension model when the problem domain was familiar (Brooks, 1983) while in cases where the problem domain was unfamiliar to the expert programmer, the bottom-up comprehension model was invoked (Pennington, 1987; Bergantz & Hassell, 1991). In other words, when the application or problem domain is familiar, expert programmers initially have a more abstract domain model before progressing to detailed program model of a program while on the other hand, when the application or problem domain is not familiar, expert programmers initially have a more detailed program model before progressing to an abstract domain model of a program.

von Mayrhauser and Vans (1995) observed a mix of the top-down and bottom-up comprehension models in some expert programmers and therefore united earlier theories of program comprehension in an integrated comprehension model where a programmer invokes three mental models: the *domain model* that incorporates problem domain knowledge; the *program model* that encompasses the program control-flow abstraction; and the *situation model* that describes data-flow and functional abstractions.

2.2.2 Mental models of novice programmers

Several studies have also focussed on mental models of novice programmers. Corritore and Wiedenbeck (1991) studied mental models of novice programmers when comprehending procedural programs. They found that mental representations of the novice programmers were primarily detailed, concrete mental representations of the program text, with little or no modeling using real-world references. In other words, the novice programmers had more of the program model other than the domain model.

Wiedenbeck and Ramalingam (1999) also studied whether mental representations of novice programmers focus more on domain-

level or program-level knowledge and whether the mental representation of object-oriented programs differ from procedural programs in a program comprehension task. The study results indicated that novices tend to develop a mental representation of small object-oriented programs strong in function-related knowledge, but weaker in data flow and program-related knowledge. On the other hand, novices' mental representations of small procedural programs were stronger in program-related knowledge. In other words, when comprehending an object-oriented program, the novices invoked more of their domain model than the program model while for the procedural program, the novices tended to invoke more of their program model other than the domain model. It thus seems that a programming paradigm can influence a dominant mental model that novice programmers can invoke.

In program authoring, Ramalingam et al. (2004) suggested that teaching programming from the object-oriented perspective, other than from a procedural perspective, may in itself assist the goal of developing the domain mental model, because the high salience of objects, their attributes, and the relationships of objects highlights the correspondence of computing objects to real world objects. However, Ebrahimi and Schweikert (2006) found that when novices are taught object oriented programming concepts at an early stage, they tend to spend more time trying to understand objects and less time on problem solving. Hence an early formation of domain mental models may not necessarily be helpful for novices particularly in the program authoring process.

Novice programmers often also have incorrect program models when learning to program. Kurland and Pea (1985) found that novice programmers sometimes have mistaken program models by having a wrong view of some programming concepts. In particular, they found that children learning to program through the LOGO programming environment had wrong mental models of the concept of recursion. Götschi, Sanders, and Galpin (2003) also found that novice programmers often held a non-viable or inappropriate *looping* program mental model of recursion, rather than the viable

or appropriate *copies* program mental model. A mental model of a programming concept is viable if it allows one to accurately and consistently represent the mechanics of a programming concept while a mental model of a programming concept is non-viable if the programmer has misconceptions about the mechanisms of the programming concept (Götschi et al., 2003). Ma, Ferguson, Roper, and Wood (2007) also found that at the completion of the first year course one third of students still held non-viable mental models of value assignment in a program, with only 17% of students holding viable mental models of reference assignment in a program. In addition, it was found that the students with viable mental models performed significantly better than those with non-viable mental models in assigned programming tasks. Jimoyiannis (2013) explored secondary education students' mental models of the concept of programming variable and the assignment statement and found that the students' thinking and application patterns were prevalently based on non-viable mathematical-like mental models about the concepts of programming variable and the assignment statement.

It is therefore important that novice programmers have correct mental models that also correspond to the task at hand and the programming paradigm in use. Thus, for example, the domain model may be appropriate in the program comprehension process in the object oriented paradigm while the program model may be appropriate in the program comprehension process in the procedural programming paradigm.

2.2.3 Techniques for characterizing programmer's mental models

There are several techniques that are used in characterizing programmer's mental models. For example, Jimoyiannis (2013) used the Structure of the Observed Learning Outcome (SOLO) taxonomy to explore novice programmers' mental models of the concept of the programming variable and the assignment statement by collecting data in the form of the novice programmers' written re-

sponses to programming tasks related to short code programs and having the responses mapped to the different levels of the SOLO taxonomy. The novice programmers in this case were secondary education students. The Structure of the Observed Learning Outcome (SOLO) taxonomy is a general educational taxonomy for classifying learning outcomes in terms of their complexity, thus enabling instructors to assess students' work in terms of its quality not of how many responses in a particular subject task or activity are correct (Jimoyiannis, 2013). The taxonomy includes five levels revealing the structural complexity of students' knowledge as they learn and the lower levels focus on quantity (the amount the learner knows) while the higher levels focus on the integration, the development of relationships between the details and other concepts outside the learning domain (the integration of the details into a structural pattern) (Jimoyiannis, 2013).

Another common technique used in studying programmers' mental models is *program summary analysis*. For example, program summary analysis has been used to characterize mental models of novice programmers (Corritore & Wiedenbeck, 1991). Program summary analysis has also been used to characterize mental models of expert programmers in program comprehension (Pennington, 1987). Hoadley, Linn, Mann, and Clancy (1996) also used program summary analysis to characterize mental models of students capable of reusing program code. Good (1999) also used the technique to describe how mental models depend on the underlying programming paradigm while O'Shea and Exton (2004) used program summary analysis to investigate how mental models depend on the task type of the programmer. Hughes and Buckley (2004) and Sajaniemi and Kuittinen (2005) also used program summary analysis to evaluate learning outcomes in novice programmers.

Program summaries can come in various forms such as transcripts of audio or videotaped interviews, written-down explanations of programs, etc. In general, program summaries allow subjects to express what they think, using their own words, at their chosen level of abstraction and detail (Good, 1999). By omitting

detailed instructions about the form of the summary, subjects' own preferences guide the selection of information in the summary and a wide variation in the responses is usually achieved. The program summary methodology avoids the problems of false positive results often associated with binary choice questions, and the difficulties in designing sensitive and reliable multiple choice questions (Good & Brna, 2004). In program summary analysis, the interest is not in the correctness of the summary; the abstraction level and the types of information are more important characterizations of the mental model than an error-free memorization of the program code.

Different techniques can be used to analyze program summaries. For example, Pennington (1987) analyzed program summaries by dividing the summaries into statements and classifying the statements into data flow, control flow or functional, Good (1999) improved Pennington's technique by analyzing program summaries using two independent measures: information types classification and object description categories. In information types classification, program summary analysis is based on what kind of information about a program each statement reveals. On the other hand, object description categories focus on the way individual objects are described in program summaries. There are seven categories in the object description categories classification, namely:

- Program only – references to objects which can occur only in the program realm.
- Program – references to objects which could be described at various levels in program terms.
- Program–real-world – references to objects which can occur in both the real-world and program realms.
- Program-domain – references to objects which can occur in both the program realm and the problem domain.
- Domain – references to objects which can occur only in the problem domain.

- Indirect reference – anaphorically referencing an object e.g. through personal pronouns.
- Unclear – objects that can not be classified due to ambiguity, lack of clarity or lack of identification.

Later, Byckling, Kuittinen, Nevalainen, and Sajaniemi (2004) evaluated the technique by Good (1999) and suggested some improvements. For example, selection of objects and detection of object references in program summaries might be difficult to agree on, particularly if there is more than one rater. Hence it is suggested that objects are agreed on first, before the categorization process.

2.3 MENTAL MODELS OF SPREADSHEET USERS

It is well documented that many spreadsheet users are not expert programmers, although the art of creating a spreadsheet itself is a programming activity (Sajaniemi & Pekkanen, 1988; Green, 1990). These non-expert programmers are actually end-users hence many spreadsheet creators can rightfully be called “end-user programmers”. Many researchers have also investigated mental models of this set of end-user programmers.

Saariluoma and Sajaniemi (1989) showed that spreadsheet users employ visual images or image-based mental representations in planning manipulations on a spreadsheet. This result is consistent with the well-known importance of visual imagery in other kinds of problem solving (Green & Navarro, 1995). Navarro-Prieto and Cañas (2001) found that the spreadsheet, with its visual characteristics, helped programmers develop a mental model of their program based on data flow structure even in the easiest tasks. This is in contrast to what programmers in traditional programming languages have. Use of imagery in spreadsheets was attributed to enhance access to data flow information. This imagery seems to have an effect in the mental models of spreadsheet users.

Mittermeier and Clermont (2002) assumed that the mental model of the spreadsheet author is influenced by layout or geometry in

spreadsheets and as such it is almost certainly line-based, column-based, or block-based. Unfortunately, geometric considerations in spreadsheets can be misleading because data and control-flow in spreadsheets are not made explicit. And to avert this problem, Mittermeier and Clermont (2002) also proposed enhancing the mental model from a geometrical based one to logical areas which offers an abstraction from the granularity of cells in form of semantic units and semantic classes. The logical areas are based on equivalence classes which are based on either on structural criteria of formulae or on equivalences with respect to usage of data.

Hendry and Green (1994) have specifically investigated mental models of spreadsheet authors. They conducted a study in which spreadsheet authors were tasked to explain their own spreadsheet. The purpose of the study was to identify general types of information that spreadsheet authors find important when explaining their spreadsheets to others. Transcripts of the interviews conducted for the study (i.e., program summaries) were analyzed and the following four information types were identified: internal and external origins of data in a spreadsheet; problem domain descriptions of data in a spreadsheet; problem domain explorations of spreadsheet data roles; and computational domain explanations of the roles of spreadsheet regions. Study findings indicated that participants were concerned with the external origins of data and described spreadsheet data in terms of problem domain constructs. Further, various roles of data in a spreadsheet were also explained in terms of the problem domain; and participants not only described regions of a spreadsheet in problem domain terms but also in terms of their computational role.

In general, many researchers have criticized the traditional spreadsheet paradigm for its low conceptual level, e.g. (Hendry & Green, 1994; B. A. Nardi, 1993; Clermont, 2005; Ayalew, 2007) and it has been argued that this low conceptual level causes many errors in spreadsheet development and use (Tukiainen, 2001). It is thus imperative that researchers explore ways of raising the conceptual level of spreadsheets. For example, Tukiainen (2001), proposed the

structured spreadsheet calculation paradigm in which implicit logical collections of cells in spreadsheets are made explicit so that the users can refer to these structures as a whole and connect these structures to computation so that the changes in structures will be reflected automatically to the computation. And as such, by offering this kind of abstraction, the conceptual level of the structured spreadsheet calculation becomes higher than the traditional spreadsheet calculation to reflect the mental model of the spreadsheet user.

2.4 MENTAL MODEL MAPPING AND PERFORMANCE

It is important for features of a system to match with the corresponding mental models of users as otherwise there will be mental conflicts resulting in errors and other suboptimal behaviour (B. A. Nardi & Zamer, 1993). Systems whose features match with the corresponding user's mental models of the same are easy to learn and use (Carroll et al., 1987).

An individual can also have simultaneous multiple mental models of a system which reflect the different levels of abstraction the person might have of a particular system when performing different tasks on the system (Carroll et al., 1987). Therefore it is important that simultaneous mental models map to each other accordingly for optimal behaviour.

In programming, Brooks (1983) theorized that, in a program comprehension task, programmers understand a completed program in a top-down manner by reconstructing knowledge about the problem domain of the program and mapping that to the actual program code itself. The mapping goes through several intermediate domains, and comprehending a program involves reconstructing part or all of these mappings. And according to von Mayrhauser and Vans (1995), program variables belong to the program mental model (program model), and they exist in order to implement some need in the application domain mental model (domain model). Therefore, meaningful variable names may ease the mapping between the program mental model and the appli-

cation domain mental model in a program comprehension task. Visual aids have also been known to improve mental models as well as their mappings because they emphasize semantic relationships that constitute the information that would facilitate better performance depending on the problem being solved and the task at hand (Navarro-Prieto & Cañas, 2001).

In spreadsheets, symbolic names and formula translation have been used with the hope to clarify the mapping between various levels of abstraction of a spreadsheet. We postpone a detailed discussion of this to Chapter 4 (“A Spreadsheet Visualization with a Mental Model Perspective”), Section 4.6.

2.5 CONCLUSION

In this chapter, we reviewed related work on mental models and spreadsheets. In particular, we looked at various perspectives on the definitions of the term “mental model” and the theory of mental models in general. We also reflected on various research works on mental models of programmers in general and spreadsheet users in particular.

3 *An Empirical Study of Spreadsheet Authors' Mental Models*²

3.1 INTRODUCTION

As presented in Chapter 1, spreadsheets are easy to create and manipulate because of the intuitiveness and simplicity of the spreadsheet interface although this simplicity comes with a cost—many spreadsheets have non-trivial errors (Panko, 1998; Powell et al., 2009). And this problem of errors in spreadsheets is compounded by spreadsheets being generally difficult to comprehend (Davis, 1996). Many tools and techniques have been developed to try to help spreadsheet authors in producing error-free spreadsheets (see Chapter 1, Section 1.1.9)

We argue that it is important to first of all understand what types of mental models spreadsheet authors possess when they are doing different spreadsheet process activities. This can help us to understand why the spreadsheet process is so error-prone and it can also help us to develop the right tools and techniques for spreadsheet activities. This approach also goes beyond the use of general user interface design principles as it provides detailed information of the specific tasks in spreadsheet work.

As stated in Chapter 1 and Chapter 2, in our context, we borrow the definition of the term “mental model” as given by Doyle and Ford (1998) where a mental model is defined as

²The condensed version of this chapter is published as a refereed paper: Kankuzi, B., & Sajaniemi, J. (2013). An empirical study of spreadsheet authors' mental models in explaining and debugging tasks. In *Proceedings of the 2013 IEEE Symposium on Visual Languages and Human-Centric Computing* (pp. 15 – 18). Washington, DC, USA.

“a mental image of the world around us that we carry in our heads depicting only selected concepts and relationships that represent real systems.”

Thus, a mental model of a spreadsheet does not carry all possible information, but just those aspects of the spreadsheet that the user finds appropriate for the task he or she has.

Any tool that a spreadsheet author uses should match the corresponding mental model as otherwise there will be mental conflicts resulting in errors and other suboptimal behaviour (B. A. Nardi & Zarmer, 1993). As the number of spreadsheet authors is vast and mixed, it is more rational to adapt spreadsheet systems to authors than to try to do the opposite. Although, one could argue that it is hard (if not even impossible) to adapt a system to a “vast and varied set of users”, we believe that understanding how a varied set of users generally thinks when using a system to perform a task could potentially offer clues on how to adapt a system to its users. Therefore, we set our research questions as follows:

RQ1 – *Do spreadsheet authors have several mental models when they are doing various spreadsheet tasks, and particularly, when they are explaining and debugging their spreadsheets?;*

RQ2 – *What are the roles of mental models in spreadsheet authors when they are doing various spreadsheet tasks, and particularly, when they are explaining and debugging their spreadsheets?;*

RQ3 – *What are the implications of mental models in spreadsheet authors on spreadsheet authoring and debugging tools?*

In this chapter, we therefore report on a study we carried out to explore the nature of mental models of spreadsheet authors when they are explaining and debugging their own spreadsheets. We will start with study design and results, followed by a discussion of the findings and implications for tool development. The chapter ends with short conclusions.

3.2 THE STUDY

In this study, we were interested to reveal the nature of spreadsheet users' mental models in two different activities: explaining a spreadsheet and debugging a spreadsheet. Consequently, the participants were first asked to explain one of their own spreadsheets, and then to find and fix seeded errors in the same spreadsheets. Talk-aloud protocols were then analyzed to reveal the nature of participants' mental models in these activities.

3.2.1 Participants

Participants were recruited through word of mouth. Twelve persons (five women and seven men) volunteered to participate in the study. The participants work in various companies and organizations in Malawi in southern Africa. Each participant is an accountant by profession with a senior managerial position. Senior managers were chosen because they are actually involved in authoring spreadsheets unlike their juniors who just use already constructed spreadsheets. Each of the participants use spreadsheets on a daily basis and all of them use Microsoft Excel spreadsheet application program for their spreadsheets. One person, a male participant, was unable to participate in the second activity due to work-place relocation; so, only the remaining eleven participants are included in the results.

3.2.2 Procedure

Before the two activities, each participant was requested to choose one spreadsheet they had created themselves. For the first activity, we printed hard copies of the spreadsheets in advance to be used in the explaining activity. For the second activity, we seeded each of the spreadsheets with errors that could be described at various levels of abstraction. For ecological validity, we used common errors that spreadsheet authors typically make (Raffensperger, 2008). The errors were not applicable to all spreadsheets but we generalized

the errors we used as follows: title label or text changed so that some basic semantics of the spreadsheet are changed; a referenced range changed in a formula cell; a formula overwritten with a constant value (input value) in a group of copy-equivalent formulae; and a formula changed by adding/subtracting a constant value.

In the first activity, explaining, each participant was requested to explain the chosen spreadsheet to the researcher. Specifically, each participant was asked to identify and circle groups of cells that “go together” (related cells) on a hard copy of their spreadsheet; thereafter participants were also requested to explain why they grouped those cells the way they did. In essence, the participants were being asked to identify different spreadsheet regions on a hard copy of a spreadsheet based on their semantic meaning. During the explanation, the researcher asked some further questions to clarify vague points. The explanation was recorded and later on transcribed accordingly.

Appendix A contains part of the spreadsheet used by participant number 2 in the explanation task. Figures and other data have been blurred in the spreadsheet to maintain confidentiality of participant’s information. Appendix B contains the transcript of the interview with participant number 2 in the explanation task. The data for participant number 2 has been randomly selected to be included in the Appendix for illustration.

In the second activity, debugging, conducted four months after the first activity, each participant was given an erroneous copy of their spreadsheet and asked to locate the errors and to fix them. The researcher told each participant beforehand that error seeding was part of the research methodology, and all participants were content with this explanation. Participants were requested to explain their activities to the researcher by talking aloud while working. Again, the explanation was recorded and transcribed later on.

In order to maintain the ecological validity of this study and to avoid the potential effects of new equipment and environment, the activities were conducted at the participants’ place of work.

3.2.3 Coding of data

The debugging activity was split into two sub-activities: locating errors and fixing each identified error. Accordingly, the transcripts of the second activity were split into error locating episodes and error-fixing episodes. Therefore, for each participant in the study, we had three transcripts: for the explaining task, the error locating task, and the error fixing task.

We adapted the program summary analysis technique of object description categories by Good (1999) in analyzing the content of each transcript as follows. We first extracted all object references that dealt with the spreadsheet application (like "cell" or "net book value") but excluded other nouns (such as "computer", "spreadsheet", etc.). Each object reference was then coded on a yes/no scale with respect to three types: "real-world", "domain-specific", and "spreadsheet-specific". We considered as real-world those object references where the object is named in terms of terminology that is specific to the real world and as such can be understood by every one (and not only by profession or industry specialists). In other words, the terminology can be understood by everyone independently of the application or problem domain of the current spreadsheet, e.g., "a car", "an airplane", etc. We classified as domain-specific those objects that are named in terms of terminology that is specific to the problem domain or application. For example, terms such as "depreciation" are specific to the accounting domain while terms such as "ticket drop-out" are specific to the airline industry. Finally, we classified as spreadsheet-specific those objects that are named in spreadsheet specific terminology, e.g., "a cell", "B1", "column", etc. It is important to note that, depending on context, an object reference might sometime fall into more than one category, e.g., "revenue", "budget", etc. might be real-world as well as domain-specific at the same time. For example, a layperson might understand what "budget" is and at the same time, "budget" is terminology that is also applicable in the accounting domain.

The coding scheme was validated by introducing it to an independent rater. Then the researcher and the independent rater coded

the transcript of one task independently and discussed the differences among their codings. Having repeated this for another transcript, they coded independently a third transcript and the inter-rater reliability was measured using Cohen's kappa in the three types. The kappa values were 0.86 for real-world, 0.69 for domain-specific, and 0.95 for spreadsheet-specific cases. As kappa values above 0.60 are considered to be good, and values over 0.80 are excellent (Stemler, 2001), the rest of the transcripts were coded by the researcher only.

Appendix C illustrates the object reference classification used in program summary analysis of transcript for participant no. 2 in the spreadsheet explanation task.

3.3 RESULTS

The spreadsheets that the participants selected for the study contained simple accounting applications and varied in size from 144 to 2691 cells (including empty cells). Table 3.1 gives a summary of the spreadsheets and the activity sessions.

Table 3.1: Summary of spreadsheets and sessions in the three tasks.

	Mean	Min	Max
Number of formula cells	201	0	504
Number of numeric value cells	140	6	288
Number of text value cells	87	28	212
Total number of cells	978	144	2691
Length of explaining sessions (min)	11:05	5:11	18:18
Length of debugging sessions (min)	13:17	4:58	27:31

The number of coded object references varied from 13 to 115 (mean 76) in the explaining task, from 16 to 499 (mean 109) in the error location task, and from 12 to 498 (mean 84) in the error fixing tasks. The relative frequencies of the three object reference types in

An Empirical Study of Spreadsheet Authors' Mental Models

the transcripts of the three tasks are given in Table 3.2 (where M stands for mean and S.D. stands for standard deviation), and are further illustrated in Figure 3.1.

Table 3.2: Relative frequencies (%) of different object reference types in the three tasks.

Obj.ref. type		Task						Total	
		Explaining		Locating		Fixing		M S.D.	
		M	S.D.	M	S.D.	M	S.D.		
Real-world	Domain	51.5	15.8	21.7	22.0	20.5	15.4	31.2	22.7
	Spreadsheet	57.9	22.9	52.2	18.8	36.8	25.0	49.0	23.5
		16.7	14.1	51.3	23.5	64.5	22.1	44.2	28.4

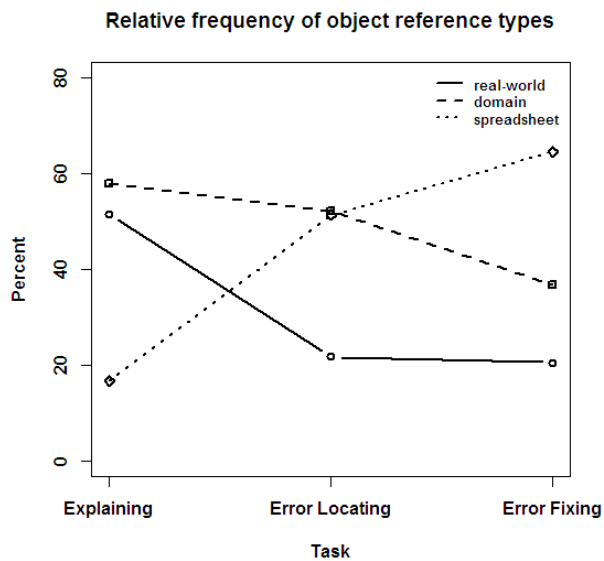


Figure 3.1: A line illustration of the relative frequencies of different object reference types in the three tasks.

To test for statistical significance of the results, we ran a two-way repeated measures analysis of variance (ANOVA) with two within-

subjects factors with the relative frequency of each object type as the dependent variable and two independent variables: the type of task with three levels (explaining, error locating and error fixing), and the type of object reference (real-world, domain-specific, spreadsheet-specific); see Figure 3.1. We used an alpha level of 0.05 for all the tests. There was no main effect either of task ($F(2,20) = 0.169$, $p = 0.846$) nor of object reference types ($F(2,20) = 2.389$, $p = 0.117$). There was, however, a significant two-way interaction of task and object reference types ($F(4,40) = 23.27$, $p < 0.0001$), i.e., differences in the frequencies of each object type are due to the joint effect of the type of task at hand and the type of object reference.

In order to study the statistical significance in more detail, for example, to compare domain-specific object references in explaining and locating tasks, one would carry out post-hoc tests. However, as the data is quite small, with large individual differences among participants, such results would be dubious. Therefore, we will make the conservative assumption that they are not and base our discussion on the overall tendencies.

3.4 DISCUSSION

We will first discuss what the overall differences in the use of different object types tell us about the role of mental models of spreadsheet authors. Then, we will look at the differences in the three tasks of explaining, error locating, and error fixing. We will first consider differences between the three tasks and then within each of the tasks individually. This is followed by more detailed examples of the role of mental models in debugging activities. Finally, we will discuss what implications our results have on spreadsheet tool design and consider the validity of this research.

3.4.1 Existence of several mental models

Research into psychology of programming, e.g., (Letovsky, 1987; Pennington, 1987; Corritore & Wiedenbeck, 1991; von Mayrhauser & Vans, 1995), has revealed that programmers have several mental

models of a program comprising such as the programming language level description or the application or problem domain level description of the same functionality. Therefore, it seems likely that spreadsheet authors have several mental models of their spreadsheets, e.g., the spreadsheet notation level model or the problem domain level model.

In the analysis of the participants' protocols, we looked at the objects that the participants referred to in their speech and at the *specific words they used in referring to those objects*. As discussed in the Literature Review chapter (Chapter 2), a similar technique has been applied in program summary analysis to characterize programmers' mental models of traditional programs (Pennington, 1987; Corritore & Wiedenbeck, 1991; Good, 1999; Hoadley et al., 1996; Hughes & Buckley, 2004; O'Shea & Exton, 2004; Sajaniemi & Kuitinen, 2005). In our study, we used it to characterize the nature of spreadsheet authors' mental models of their spreadsheets, that is, to what extent they understand a spreadsheet in real-world terms, in domain-specific terms, and in spreadsheet-specific terms. Furthermore, we looked at the differences in the mental models in various tasks: explaining, error location and error fixing.

The results showed that even though the participants used overall roughly the same amount of the three object types, their use varied significantly between the three tasks of explaining, error locating and error fixing. This means that the participants had different ways of thinking about a spreadsheet and that they were able to change their viewpoint depending on the task at hand. Thus, in essence, their mental models of the spreadsheets contained all these elements and they were able to concentrate on some specific components of their mental models depending on the nature of the task. We hypothesize that these elements can be characterized as three mental models: the *real-world model* that comprises general knowledge of the world around us; the *domain model* that represents knowledge of the application or problem domain and the functionality of the spreadsheet in application or problem domain terms; and the *spreadsheet model* that codes the expressions and data rela-

tionships in the spreadsheet.

We do not argue that these mental models would cover all the knowledge that is needed in working with spreadsheets; neither do we claim that the division of this knowledge into the three models is definite. However, we do argue that there are conceptually varying types of spreadsheet knowledge and that spreadsheet authors use different types of knowledge in different tasks. As these varying types of knowledge depict selected concepts and relationships that represent a spreadsheet, we refer to them as mental models (Doyle & Ford, 1998). In the following, we look more closely at their use in different tasks, and contemplate their role in devising methods and tools for different activities in spreadsheet creation, testing, debugging, and use.

3.4.2 Mental models between different tasks

The results indicate that during each of the tasks, participants think multi-dimensionally with different activation levels of each mental model at the same time. We thus codify the results that were illustrated in Figure 3.1 in Table 3.3.

Table 3.3: Codification of nature of mental models of participants as they performed the three tasks of explaining, locating errors, and fixing located errors in their spreadsheets.

Mental Model	Explaining	Error Locating	Error Fixing
Real-world model	high	low	low
Domain model	high	high	medium
Spreadsheet model	low	high	high

Referring to Table 3.3, the *real-world model* significantly diminishes when performing debugging tasks (locating and fixing errors) unlike when explaining a spreadsheet. It seems that explaining a spreadsheet includes a description of the connection between real-world concepts and the spreadsheet; in debugging the spreadsheet is treated in a more isolated fashion. Thus, unlike when explain-

ing a spreadsheet, real-world constructs are used minimally when performing debugging tasks.

Overall, the *domain model* is the most prominent mental model with the relative frequency of 49.0%. There are no significant differences in the role of the *domain model* when a participant is explaining a spreadsheet or locating errors in a spreadsheet, but its role diminishes a little when they are fixing errors in a spreadsheet.

The *spreadsheet model* is quite low when explaining a spreadsheet unlike when doing the debugging tasks. One might argue that the increased use of spreadsheet terms in debugging is due to the lack of domain-specific names of individual cells, but this is not the case, because all cells represent problem domain data items and can hence be named in problem domain terms. We thus hypothesize that to explain a spreadsheet, authors concentrate on the problem domain content, but to locate errors as well as to fix located errors, one must also be well conversant with technical intricacies of a spreadsheet such as cell referencing, hence the spreadsheet model becoming more pronounced in the debugging tasks.

3.4.3 Mental models within different tasks

Looking at the three mental models simultaneously for each given task as illustrated in either Figure 3.1 or Table 3.3, it is clear that when *explaining* a spreadsheet, the participants talk (i.e., think) mainly in terms of problem domain and real-world concepts, but the spreadsheet model seems to be less evident. It is also clear that when *locating errors*, study participants think mainly in terms of problem domain and spreadsheet concepts while the real-world model seems to be rather less evident. This is unlike in the explanation task where both the domain and real-world models are prominent. The relationships between mental models in the explaining task and error locating task are illustrated in Figure 3.2.

We hypothesize that during the explanation task, real-world concepts are mainly related with problem domain concepts with little reference to the spreadsheet model. Thus, an explanation of a spreadsheet is more of an explanation of its functionality in the

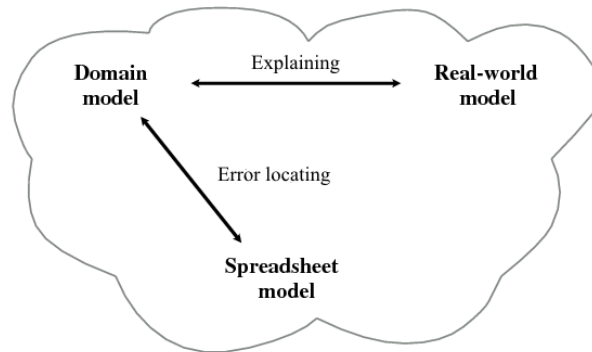


Figure 3.2: An illustration of the relationships between mental models in the explaining task and error locating task.

application or problem domain than a description of its implementation in the underlying spreadsheet system. In the error locating task, participants think mainly in terms of problem domain and spreadsheet concepts, that is, they are mainly concerned with checking if problem domain concepts map correctly to spreadsheets concepts and vice-versa. Thus, there is a noteworthy difference in the behavior of the participants in the two tasks of explaining and error locating.

When *fixing errors*, the situation is different: there is now a single major mental model (as opposed to two major models in both explaining and error locating tasks): the spreadsheet model. In order to understand the reason for this, let us first have a look at different error types.

An error occurs if there is a mismatch between what is in the spreadsheet and what is in the application domain or real-world. A mismatch between, for example, the problem domain and the spreadsheet can be due to an error in the domain model or due to an error in the mapping between the domain model and the spreadsheet model; we will call such errors “domain errors”. It is also possible that an error occurs within the spreadsheet model only. There are thus three types of errors: domain errors, real-world errors, and spreadsheet errors. For example, a mistake in remem-

bering the number of days in a year is an error in the real-world model; picking up wrong values for net book value calculation is an error in the domain model; entering a wrong cell reference is an error in the spreadsheet model. Of course, it is not that easy to see the reason for a bug by looking at the bug only (e.g., the latter two errors may show up as the same bug), but the important point is to realize that errors may be related to any of the three mental models.

In fixing domain errors, we could repeatedly identify spreadsheet authors correcting the error in the domain model (in their head) and thereafter quickly switching back to the spreadsheet model to correct the error in spreadsheet-specific terms to reflect the correct domain model. In the case of spreadsheet errors, they correct the error in spreadsheet terms and thereafter verify that the correction also reflects correctly the domain and real-world models.

Thus, the roles of the three mental models in the error fixing task can be explained by the different nature of individual errors: fixing domain-related errors requires the mapping between domain and spreadsheet concepts; fixing real-world-related errors requires the mapping between real-world and spreadsheet concepts; fixing spreadsheet-related errors requires spreadsheet concepts only. Thus one would expect to see more spreadsheet concepts than problem domain and real-world concepts during the fixing task. We illustrate the relationships between mental models in the error fixing task in Figure 3.3.

3.4.4 Mental models in debugging episodes

The previous subsections considered the role of mental models between and within the three tasks of explaining, error locating and error fixing. We will now look in more detail the role of mental models in debugging, i.e., locating and fixing errors, by looking at participants' behavior in debugging three seeded errors—one for each error type.

One example of a seeded *real-world error* involved erroneously calculating the distance covered by a vehicle by subtracting a larger

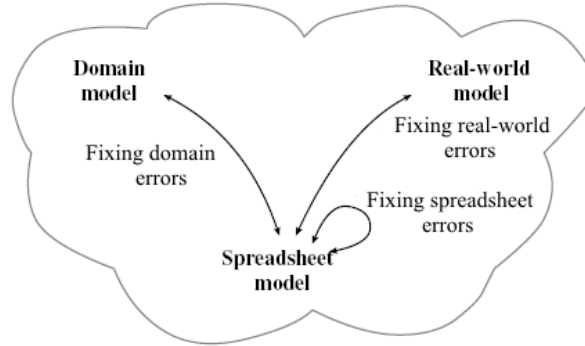


Figure 3.3: An illustration of mapping of mental models in the error fixing task.

figure from a smaller figure which would give a negative covered distance. A negative covered distance certainly can not make sense in the real world. The very same error could be described in the domain model as covered distance being erroneously calculated from “opening mileage minus closing mileage” instead of being calculated from “closing mileage minus opening mileage”. In the spreadsheet model, the error could be achieved by changing a formula, say, from $B46=B44-B43$ to $B46=B43-B44$ and thereafter applying the same to the whole row 46 which is the covered distance.

The following is an excerpt from a transcript of an interview with participant number 8 whose spreadsheet was seeded with this error:

[In the middle of error locating:] ... I will also check the covered distance, see if is opening, closing minus opening I see that it's not... then I will have to make sure that it reads as how they are supposed to, it reads closing minus opening and this should be the same thing for all other areas..... but then I see that there is a problem here on column O, we've got minus and also on N, I need to find out why we have got a minus value and I see there is a problem in the closing mileage [Error fixing starts:] which I need to double check on my other records to

see why we have got a smaller number on the closing mileage then I will put ... mark it yellow for following up, same thing on cell O44, I will put a yellow thing there... mark, same thing on cell P4 ahah P44, ... okay [Error locating continues:] but the rest of them are okay because of the fact that the figures are positive and also of course on E44 probably the figure has not yet been recorded there that's why there is a zero and also on D44 which I intend to follow up...

In the preceding excerpt, the participant firstly identifies the error in the domain model and also corrects the error in the domain model before effecting the correction in the spreadsheet model on the spreadsheet on all cells in the affected row. The participant also verifies the effected corrections against what is also in the real-world model. Suspect problematic cells are marked for further following up after double checking with physical records.

An example of a seeded *domain error* was to change a "gross contribution" calculation in one spreadsheet from "Total Revenue – Total Direct Operating Costs" to "Total Revenue + Total Direct Operating Costs". In the spreadsheet model, we had a formula changed from $F29=F13-F27$ to $F29=F13+F27$. The following is an excerpt from a transcript of an interview with participant number 2 whose spreadsheet was seeded with this error:

[In the middle of error locating:] Let me check subtotals.....This gross contribution is supposed to be... total revenue less total operating costs but as you can see I have just highlighted the total contribution for this column it's showing as you see it's a plus...in that formula it's like these are also revenues yet they are expenses [Error fixing starts:] so we can correct the formula...this formula should be minus...let me check...it's because this is 600 something that's why it's like this, this one it's the same so I will copy this formula to here...also this one it's the same I copy this formula because it was added

instead of subtracted. Sure...uuuuuh...

In the preceding excerpt, the study participant is primarily describing the error in the domain model. The error is also corrected in the domain model before being effected in the spreadsheet model on the spreadsheet.

An example of a seeded *spreadsheet error* we had in this study, was changing the referenced range in a formula cell. For example, we changed the range in the formula in cell H36 of the spreadsheet of study participant number 6. A formula $H36=H15+H28+H34$ was changed to $H36=H28+H34$ hence H15 was deliberately excluded. In the domain model, this very same error could be described as: "The grand totals are excluding furniture and fittings". An excerpt of the transcript for this error is as follows:

[In the middle of error locating:] ... then I can check on the grand total, grand total we have a problem, grand total there is an error, [Error fixing starts:] grand total should be equal to the total for furniture and fittings plus total for computers and office equipment and total for motor vehicles, so I will also change....so I have to change I have to correct, I have to correct by including all the totals... [Error locating continues:] so I can check the rest of the totals ...it shows that all the totals... we have errors in all the totals.. grand totals ... [Error fixing continues:] so it means I can use this corrected grand total for opening balance to correct all the grand totals for the rest of the figures so it means I have corrected all the errors I believe that were here...

Again, in the preceding transcript excerpt, the participant firstly identifies the error in the domain model and subsequently makes a correction in the domain model before effecting the change in the spreadsheet model. Thereafter, a corrected formula, although still being described in the domain model, is copied to other similar parts of the spreadsheet ("I can use this corrected grand total for

opening balance to correct all the grand totals for the rest of the figures").

Looking at the examples of the seeded errors, it is clear that an error can be described in more than one way: the same error can be described in the real-world model, the domain model as well as in the spreadsheet model. However, this study indicates that errors are primarily described in the domain model while also being mapped to the spreadsheet model such as cell references.

3.4.5 Implications for tool development

There are various tools to aid in the comprehension of spreadsheets as well as in debugging spreadsheets. However, results of this study indicate that there have to be some pre-requisite characteristics of these tools if they are to be effective in their intended purpose. For example, understanding how spreadsheet authors explain their spreadsheets is important because explaining what a spreadsheet does is closely linked to comprehending and modification of a spreadsheet (Hendry & Green, 1994). Results of this study indicate that when explaining a spreadsheet, the real-world model and domain model are prominent. Therefore, a tool intended to aid in comprehension of a spreadsheet should make prominent real-world and application or problem domain concepts in the spreadsheet and map those concepts easily to spreadsheet-specific details such as cell references. Borrowing from the example given in Section 3.4.4 above, displaying a formula as "H36=H15+H28+H34" might not be easier to understand than to say "grand total = total for furniture and fittings + total for computers and office equipment + total for motor vehicles" whereby "grand total" maps to cell reference H36, "total for furniture and fittings" maps to cell reference H15, etc. In other words, spreadsheet comprehension tools should not focus much on amplifying spreadsheet specific details such as cell references because doing so will cause a mismatch with how spreadsheet users think.

Similarly, spreadsheet debugging tools cause a mismatch with how spreadsheet users think if they focus much on amplifying pos-

sible spreadsheet errors in more spreadsheet specific detail such as cell references. The results of this study indicate that to locate and fix an error in a spreadsheet, one must constantly switch back and forth between the domain model and spreadsheet model, which requires frequent use of the mapping between application or problem domain concepts and their spreadsheet model counterparts. However, as found by Hendry and Green (1994), mapping cell and range references into problem domain interpretations is difficult, and as such, it is imperative that a good debugging tool will make prominent the mapping between problem domain concepts and spreadsheet-specific details in the spreadsheet.

In other words, a good debugging tool must not only display possible spreadsheet errors in spreadsheet terms (e.g., cell references) but also in problem domain terms in order to help spreadsheet users discern errors with a more pronounced mapping between the application or problem domain and the spreadsheet. It is not enough to have a debugging tool that simply displays possible errors in terms of cell references or the underlying spreadsheet data flow without providing a corresponding mapping to application or problem domain related details of the spreadsheet. Again, borrowing from the example given in Section 3.4.4 above, it might be easier to discern that “covered distance = opening mileage – closing mileage” is erroneous than to just to have “B46=B44-B43”. This lack of mapping between application or problem domain concepts to spreadsheet specific details in many debugging tools can be one of the reasons why many apparently very good debugging tools are rarely used by spreadsheet users (Y. Chen & Chan, 2000a).

3.4.6 Threats to validity of study

The small number of participants as well as their narrow occupational and geographical distribution poses a risk of validity on this research. On the other hand, even though the findings are new they can be explained in ways that are consistent with previous literature. Moreover, the findings could be expressed in general terms that are not tied to any profession or geographical region.

The spreadsheets that the participants gave to this study were quite simple, and the participants may have given out spreadsheets which are not so crucial in their work due to privacy issues; it might also be the case that spreadsheet authors mentally process larger spreadsheets in qualitatively different ways. However, in practice most spreadsheets are simple (Sajaniemi & Pekkanen, 1988), so we can expect that the findings apply to a large body of spreadsheet usage.

3.5 CONCLUSION

This chapter reported an empirical study carried out to explore the nature of mental models of spreadsheet authors when they are explaining and debugging their own spreadsheets. Consequently, the participants were first asked to explain one of their own spreadsheets, and then to find seeded errors in the same spreadsheets. Talk-aloud protocols were then analyzed to reveal the nature of participants' mental models in these activities.

The study findings indicate that spreadsheet authors have (at least) three mental models of a spreadsheet: the real world model that comprises general knowledge of the world around us; the domain model that represents knowledge of the application or problem domain and the functionality of the spreadsheet in problem domain or application terms; and the spreadsheet model that codes the expressions and data relationships in the spreadsheet. These findings answer the research question **RQ1** – *Do spreadsheet authors have several mental models when they are doing various spreadsheet tasks, and particularly, when they are explaining and debugging their spreadsheets?*

The study findings also indicate that the roles of the identified mental models vary depending on the task at hand – the authors explain their spreadsheets mainly in terms of real world and application or problem domain concepts, i.e., they mainly use the real-world and domain mental models when explaining a spreadsheet; in debugging, they constantly switch between application or prob-

lem domain concepts and spreadsheet-specific concepts to locate errors in a spreadsheet, i.e., they mainly switch between the domain model and spreadsheet model when locating errors in a spreadsheet. In turn, they mainly use spreadsheet-specific concepts to fix an identified error, i.e., they mainly use the spreadsheet model when fixing errors. These findings answer the research question **RQ2** – *What are the roles of mental models in spreadsheet authors when they are doing various spreadsheet tasks, and particularly, when they are explaining and debugging their spreadsheets?*

The findings stated above provide insights on the need for developing spreadsheet authoring and debugging tools that correspond to spreadsheet authors' mental models given the task at hand and thus answering research question **RQ3** – *What are the implications of mental models in spreadsheet authors on spreadsheet authoring and debugging tools?* For example, the results indicate that to locate and fix an error, one must constantly switch back and forth between the domain model and the spreadsheet model, which requires frequent use of the mapping between application or problem domain concepts and their spreadsheet model counterparts. It is therefore not enough for a tool to visualize spreadsheet data flow in spreadsheet terms without providing a corresponding mapping to problem domain concepts. We present in the next chapter (Chapter 4), a possible way on how to make this information available to spreadsheet authors.

4 A Spreadsheet Visualization Tool With a Mental Model Perspective³

4.1 INTRODUCTION

Based on the results of the empirical study reported in Chapter 3, we developed an interactive spreadsheet visualization tool whose purpose is to ease the mapping between the domain/real-world mental models and the spreadsheet mental model in spreadsheet authors when they are working with a spreadsheet created originally by themselves or by someone else. We had the following research question in mind when developing the tool:

RQ4 – *Is it possible to develop a spreadsheet understanding and debugging tool that relieves users from spreadsheet details and lets them utilize more of their mental model of the application domain?*

In this chapter, we give a description of the tool.

4.2 TOOL OVERVIEW

To offer spreadsheet details in application or problem domain terms, the tool displays formulae in higher level symbolic names or problem domain/real-world terms and referenced cells are automatically highlighted. The problem domain/real-world information is extracted from labels (headers) through spatial layout information of each corresponding input cell in the formula. We call this

³The condensed version of this chapter is published as a poster paper: Kankuzi, B., & Sajaniemi, J. (2014). A domain terms visualization tool for spreadsheets. In *Proceedings of the 2014 IEEE Symposium on Visual Languages and Human-Centric Computing* (pp. 209 – 210). Washington, DC, USA.

displayed problem domain/real world information a “domain/real world narrative” or simply a “domain narrative”. We thus call this tool a domain terms spreadsheet visualization tool.

The tool has five distinguishing features and characteristics. First, the tool displays a formula as a domain/real-world terms narrative in a box just right below the active formula cell. For example, in the spreadsheet depicted in Figure 4.1, a formula in cell C9 given as “=SUM(C5:C8)” in spreadsheet-specific terms is automatically translated to “SUM(Jan | James Bourne ... Jan | Jasmine Hunt)” in domain terms. For each referenced cell, its corresponding domain/real-world terms are extracted from its column and row labels. The column and row names are separated by “|”; the naming obeys a “column | row” direction following the standard spreadsheet convention. Narratives are positioned a little bit lower than the active formula cell to avoid distractions when navigating through the cells.

The image shows a spreadsheet interface. At the top, the formula bar displays the formula `=SUM(C5:C8)` for cell C9. Below the spreadsheet grid, a cyan-colored box contains the domain narrative: `SUM(Jan | James Bourne ... Jan | Jasmine Hunt)`. The spreadsheet data is as follows:

	B	C	D	E	F	G	H	I
1	SALES REPORT							
2		Jan	Feb	Mar	Q1	Apr	May	Jun
3								
4	Region: North							
5	James Bourne	4,521	2,863	3,802	11,925	5,698	4,137	3,627
6	Chris Hewitt	3,510	4,882	3,915	12,056	3,843	1,413	4,187
7	Pat Hill	5,213	4,557	2,187	10,562	1,070	4,424	1,240
8	Jasmine Hunt	4,175	2,681	2,257	9,112	3,588	4,666	3,666
9	Total North	17,419	14,983	9,904	43,655	14,199	14,640	12,720
10								
11								
12	Region: South							
13	Mark Watts	3,400						3,915
14	Jane Hill	3,608						4,187
16	Roger West	2,359	5,273	5,438	13,070	2,280	4,882	3,915
17	Gill Smith	4,487	2,638	5,100	12,225	1,205	4,557	2,187
18	Total South	16,157	19,415	23,634	59,206	10,993	17,697	15,355

Figure 4.1: Visualization of a Sales Report spreadsheet – The cell cursor is in C9, whose formula is visible at the formula bar and narrated in domain terms in the cyan coloured box. Cells referred to in the domain narrative are highlighted with cyan background. All formula cells are marked with a magenta right border.

Second, to track changes automatically, the narratives are automatically re-generated as one works through the spreadsheet thus

providing rapid feedback to a spreadsheet author. Automatic generation of visualizations is important because users are not comfortable with tools that require much user intervention (Sajaniemi, 2000). Providing rapid feedback to the user is also one of the ideal characteristics of an end-user programming environment (Seta, Ikeda, Kakusho, & Mizoguchi, 1997).

Third, referenced cells are automatically highlighted and their background colour matches with the background colour of the narrative. For example in Figure 4.1, “Jan | James Bourne” or C5, “Jan | Chris Hewitt” or C6, “Jan | Pat Hill” or C7, and “Jan | Jasmine Hunt” or C8 are all highlighted in cyan colour. We have thus applied the Gestalt Law of Similarity whereby objects of the same colour are naturally perceived as related (Wertheimer, 1938). In this case, matching background colours of the narrative and highlighted cells helps the user to automatically perceive the two as related.

Fourth, all formula cells are marked with a magenta right border so that a user can have a general overview of the structure of a spreadsheet and differentiate formula cells from text cells as well as input number cells. For example, in Figure 4.1, cell E5 is not a formula cell while cell E9 (with a magenta right border) is a formula cell. We have thus applied cognitive theories of visual search (Desimone & Duncan, 1995) to provide a minimal clutter pop-out effect that results in better speed and accuracy in formula detection than the attentional template approach utilized in normal spreadsheet systems.

Fifth, the tool is also superimposed on the spreadsheet display. This is important because users may find it tedious and confusing to determine the correspondence between a separate visualization and the spreadsheet itself (Davis, 1996).

4.3 LABEL EXTRACTION IN THE TOOL

The problem domain/real world information displayed in the narratives is extracted from labels through spatial layout information of each corresponding input cell in the formula as the spatial ar-

rangement of the labels and formulae in a spreadsheet is typically strongly determined by the intended computation semantics (Jannach et al., 2014). For each input cell, its corresponding application or problem domain terms are extracted from its corresponding column and row labels. For labels spanning multiple cells, the tool break offs once it encounters non-textual information to have a “multiple cell label” domain term. The column and row labels are separated by “|”. The naming follows a “column | row” direction because spreadsheet cell naming follows the column-row convention. Thus a combination of a column label and a row label forms a symbolic name for each input cell.

The tool also handles situations where an input cell has an incomplete symbolic name as follows:

- (i) Missing column name – An input cell which does not have its corresponding column name has its row name as its symbolic name as illustrated in Fig 4.2.
- (ii) Missing row name – An input cell which does not have its corresponding row name has its column name as its symbolic name as illustrated in Figure 4.3.
- (iii) Missing both labels – An input cell that does not have both its corresponding row name and column name has “unnamed” as its symbolic name as illustrated in Figure 4.4.

4.4 IMPLEMENTATION

The tool is implemented as an add-in to a popular spreadsheet system, Microsoft Excel, so that it should not be completely different from the traditional spreadsheet environment. This was to take advantage of existing user experience as their familiarity with existing tools and techniques could become worthless if the tool is too different from what they know and how they are used to work (Kulesz, 2011). Experience is an asset, and learning new tools is hard. Even if one manages to convince end-users of a new tool’s benefits, they

A Spreadsheet Visualization Tool With a Mental Model Perspective

B6		fx		=B4+B5	
	A	B	C	D	E
1	BALANCE SHEET (Dollars in Thousands)				
2					
3	Current assets				
4	Cash	373			
5	Inventories	743			
6	Total current assets	\$ 1,116			
7					
8	Other assets				
9	Property and plant at cost	10,963			

Cash + Inventories

Figure 4.2: An example of a domain narrative when a column name is missing. B4 is translated to “Cash” and B5 to “Inventories” and hence the formula in B6 is translated as “Cash + Inventories”.

E5		fx		=SUM(B5:D5)	
	A	B	C	D	E
1					
2	Sales by District				
3					
4		District 1	District 2	District 3	Total
5		4000	5000	3500	12500
6		4550	5660	5670	15880
7					
8					
9					
10					
11					
12					

SUM(District 1 ... District 3)

Figure 4.3: An example of a domain narrative when a row name is missing. B5 is translated to “District 1” and D5 to “District 3” and hence the formula “=SUM(B5:D5)” in E5 is translated as “SUM(District 1 ... District 3)”.

F7		fx		=F5+F6	
	A	B	C	D	E
1					
2	Sales by District				
3					
4		District 1	District 2	District 3	Total
5		4000	5000	3500	12500
6		4550	5660	5670	15880
7					
8					
9					
10					
11					
12					

unnamed + unnamed

Figure 4.4: An example of a domain narrative when both row name and column name are missing. F5 is translated to “unnamed” and F6 to “unnamed” and hence the formula “=F5+F6” in F7 is translated as “unnamed + unnamed”

will still have to learn to cope with it in practice. As such, their experience with existing tools and techniques could become worthless if the tool is too different from what they know and how they are used to work (Kulesz, 2011; Jannach et al., 2014). Therefore, we took effort to make sure that the tool should be available in the traditional spreadsheet environment and not be completely different from it. The tool, implemented using Visual Basic for Applications (VBA), has 1227 lines of code and has been tested to run in Microsoft Excel 2003, Microsoft Excel 2007 and Microsoft Excel 2010. The tool is freely downloadable online⁴.

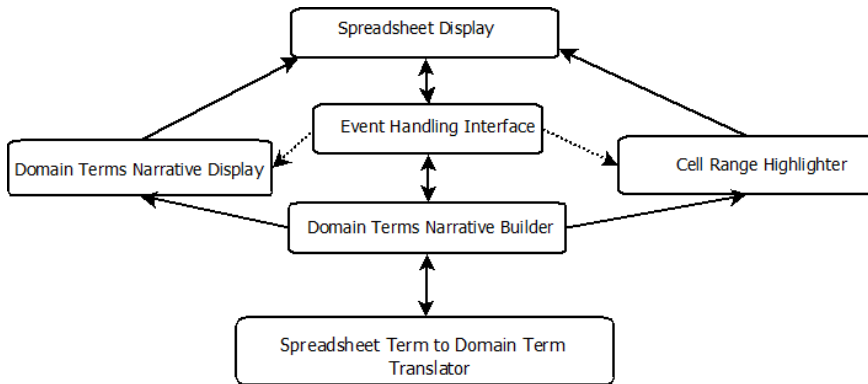


Figure 4.5: An illustration of a high-level architecture of the domain terms spreadsheet visualization tool

Figure 4.5 depicts a high-level architecture of the tool. In the tool, events such as clicking on a formula cell on the spreadsheet display result in the display of an already generated domain terms narrative as well as highlighting of its corresponding cell range. Making changes to a formula cell or a label triggers the rebuilding of all domain terms narratives for the spreadsheet with the translator doing the conversion of spreadsheet specific terms to domain/real-world terms.

⁴The tool is freely available at http://cs.uef.fi/spreadsheet_tools/

4.5 POTENTIAL USES OF THE TOOL

The tool has the potential to be used in the following spreadsheet activities:

- (i) When creating or editing a spreadsheet, the tool can help one in tracking effected changes as active formula cell ranges are highlighted and a corresponding problem domain description is displayed for an active formula cell. This has potential in helping to create an error free spreadsheet as it will be providing an intuitive on-spot feedback to the spreadsheet author on changes being done to the spreadsheet. This feedback can thus act as a verification aid.
- (ii) The tool may also be helpful in the comprehension of a spreadsheet created by others through visual comprehension aids such as highlighting all formulae in a spreadsheet and for each formula cell, a click on it will highlight its precedent cells (formula range) and a corresponding problem domain based description of the cell is given in the formula cell caption box.
- (iii) The tool may also help in the comprehension of a spreadsheet as it can give a user a general overview of the structure of a spreadsheet through the magenta right border markings on all formula cells.
- (iv) The tool may also be helpful in locating errors in a spreadsheet. For example, a problem domain description or narrative for a given formula cell, that does not match with what is expected in the problem domain could provide a visual cue for an error in that formula cell.

4.6 COMPARISON WITH OTHER TOOLS

In spreadsheets, symbolic names and formula translation have been used with the hope to clarify the mapping between various levels of abstraction of a spreadsheet. For example, many commercial

spreadsheet systems such as Microsoft Excel and Google Spreadsheets have a feature known as “named ranges”. Named ranges is a feature that allows one to assign a symbolic name to a cell or a group of cells in a spreadsheet (Google Inc., 2013). For example, instead of using, say, “A1” to designate a cell or say, “A1:B2” to designate a group of cells, one can name a cell or group of cells as say, “budget_total” or whatever title they choose. Range names are touted to allow spreadsheet authors to create meaningful categories in their spreadsheets; are also touted for making formulae more comprehensible; and also allow entering of multiple cell addresses and as such one can enter the corresponding range names. For example, instead of entering =SUM(A1:B2 , D4:E7), one can enter the simpler, more intuitive, =SUM(budget_total , quarter2). The named range feature of these commercial spreadsheets primarily differs from our tool in that in our tool we automatically generate symbolic names for spreadsheet-specific cell references instead of having a user enter them manually as is the case with the named range feature of these commercial spreadsheet systems.

Symbolic names and formula translation have also been used in some spreadsheet visualization tools. Spreadsheet Professional, a spreadsheet add-in developed by Spreadsheet Innovations Ltd. (2013), translates formulae whereby references are replaced by symbolic names obtained by searching for the first textual cell leftwards (or optionally rightwards) and/or upwards (or downwards) from the referenced cell. The drawback with finding symbolic names by just searching for the first textual cell (leftwards, etc.) is that it leaves out the neighbouring text which might form a “whole” symbolic name as in many spreadsheets, symbolic names may occupy more than one cell. In general, the notations in the translations also differ from our tool. For example, Spreadsheet Professional translates the formula, “=SUM(C5:C8)” in the active cell given in cell C9 in the spreadsheet captured in Figure 4.6 as “C9: Total North.Jan = SUM(James Bourne.Jan:Jasmine Hunt.Jan)”. In our tool, we translate the same as “SUM(Jan | James Bourne ... Jan | Jasmine Hunt)” in domain terms. Our tool also differs from the Spreadsheet

Professional tool in the spatial location and display of the domain narratives on the spreadsheet. For example, Spreadsheet Professional provides a translation of the current formula into English in a translation bar at the top of the whole spreadsheet as depicted in Figure 4.6 while we provide the same in a cyan coloured box just below the active formula cell. In our case, once a person clicks on a formula cell, the referenced cells are also automatically highlighted, this is also not the case with Spreadsheet Professional.

SP Build ▾ Test ▾ Document ▾ Use ▾ Other ▾								
Translation bar C9: Total North.Jan =SUM(James Bourne.Jan:Jasmine Hunt.Jan)								
Custom Toolbars								
C9		=SUM(C5:C8)						
	B	C	D	E	F	G	H	I
1	SALES REPORT							
2		Jan	Feb	Mar	Q1	Apr	May	Jun
3								
4	Region: North							
5	James Bourne	4,521	2,863	3,802	11,925	5,698	4,137	3,627
6	Chris Hewitt	3,510	4,882	3,915	12,056	3,843	1,413	4,187
7	Pat Hill	5,213	4,557	2,187	10,562	1,070	4,424	1,240
8	Jasmine Hunt	4,175	2,681	2,257	9,112	3,588	4,666	3,666
9	Total North	17,419	14,983	9,904	43,655	14,199	14,640	12,720
10								
11								
12	Region: South							
13	Mark Watts	3,400	5,329	4,734	13,463	1,450	4,191	3,109
14	Jane Hill	3,608	4,922	4,711	13,241	1,463	1,204	2,342
16	Roger West	2,359	5,273	5,438	13,070	2,280	4,882	3,915
17	Gill Smith	4,487	2,638	5,100	12,225	1,205	4,557	2,187
18	Total South	16,157	19,415	23,634	59,206	10,993	17,697	15,355

Figure 4.6: A translation of the formula in the active cell, C9, in Spreadsheet Professional.

D. Nardi and Serrecchia (1994) developed a spreadsheet visualization tool that also does formula translations. In this tool, labels for input cells to a formula are given out as symbolic names as illustrated in Figure 4.7 in which cell C4 is translated as “PROFIT(year_2)” and are displayed in an interface separate from the native spreadsheet system. Our approach is similar to this although we differ in the spatial location of the generated explanations (in our tool domain narratives are displayed as part of the native spreadsheet system) and we also differ in the way labels spanning multiple cells are handled when forming symbolic names. In our case, we break off once we encounter a non-textual information.

Moreover, our tool also differs from this tool in the syntactical notation of the generated explanation. For example, if in Figure 4.7, the formula in cell B4 was =B2-B3, the tool by D. Nardi and Serrecchia (1994) would generate an alternative corresponding explanation as "REVENUE(year_1) - COST(year_1)". In our case, we would translate =B2-B3 as "year_1 | REVENUE - year_1 | COST" in domain terms.

	A	B	C	D
1		year_1	year_2	
2	REVENUE		↑	
3	COST			
4	PROFIT	←		
5	ROS			
6				

Figure 4.7: An illustration of how a symbolic name is assigned to a cell in the tool by Nardi and Serrecchia (1994) - cell C4 is referred as "PROFIT(year_2)".

The screenshot shows a spreadsheet window with the following data:

SALES REPORT									
	Jan	Feb	Mar	Q1	Apr	May	Jun	Q2	Jul
Region: North									
James Bourne	4,521	2,863	3,802	11,925	5,698	4,137	3,627	13,462	5,100
Chris Hewitt	3,510	4,882	3,915	12,056	3,843	1,413	4,187	9,443	2,021
Pat Hill	5,213	4,557	2,187	10,562	1,070	4,424	1,240	6,734	4,128
Jasmine Hunt	4,175	2,681	2,257	9,112	3,588	4,666	3,666	11,920	5,410
Total North	17,419	14,983	9,904	43,655	14,199	14,640	12,720	41,559	16,659
Region: South									
Mark Watts	3,400	5,329	4,734	13,463	1,450	4,191	3,109	8,750	2,097
Jane Hill	3,608	4,922	4,711	13,241	1,463	1,204	2,342	5,009	3,402
Roger West	2,359	5,273	5,438	13,070	2,280	4,882	3,915	11,077	5,498
Gill Smith	4,487	2,638	5,100	12,225	1,205	4,557	2,187	7,949	4,043
Total South	16,157	19,415	23,634	59,206	10,993	17,697	15,355	44,045	16,518

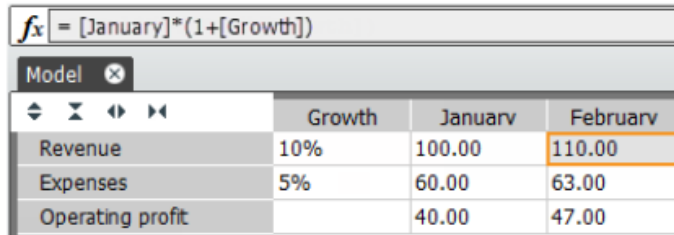
Figure 4.8: A translation of the formula in the active cell, C9, in Spreadsheet Detective.

Spreadsheet Detective, a spreadsheet add-in developed by Southern

Cross Software (2013), also translates spreadsheet formulae into a higher level form as illustrated in Figure 4.8. In Figure 4.8, the formula “=SUM(C5:C8)” in cell C9 has been translated to “C9 – ‘Jan’TotalNorth - 17419. – = SUM(C5:C8‘JamesBourne:‘JasmiHunt#)”. On the other hand, our tool translates the same as “SUM(Jan | James Bourne ... Jan | Jasmine Hunt)” in domain terms.

The SUMWISE extension to traditional spreadsheets by Miller, Miller, and Parrondo (2010) employs user-defined row and column names, rather than the usual A1 (or R1C1) references found in traditional spreadsheets. An illustration of the SUMWISE spreadsheet extension is in Figure 4.9 where the selected cell in the “Revenue” row contains the formula “=[January]*(1+[Growth])”. The SUMWISE extension differs from our tool in several ways. First, it completely replaces the traditional spreadsheet environment since traditional cell references are completely replaced with user-defined row and column names which might make the operating environment so unfamiliar to the user. In our case, we have tried to move around this challenge by presenting a formula in domain specific terms using symbolic names as well as also giving the corresponding formula in spreadsheet specific terms right in the traditional spreadsheet environment. For example, in Figure 4.1, a formula given in domain terms as “SUM(Jan | James Bourne ... Jan | Jasmine Hunt)” has its corresponding spreadsheet specific formula still displayed as “SUM(C5:C8)” in the formula bar, i.e., as part of the “standard” spreadsheet environment. In our tool, the domain formula is displayed as a narrative in a box just right below the active formula cell and not in the formula bar as in the SUMWISE extension. And in addition to this, we also clearly highlight all the referenced cells of an active formula cell. Secondly, in the SUMWISE extension, the symbolic names of cell references seem to be replaced by column headers only. We feel this might cause confusion when accessing some cells produces the same narrative. For example, in Figure 4.9, where the selected cell in the “Revenue” row contains the formula “=[January]*(1+[Growth])”, accessing the cell just below it in the

“Expenses” row would display “=[January]*(1+[Growth])”. In our case, each cell is uniquely identified by a symbolic name generated from the column header and row header. In our case we would therefore have “January | Revenue *(1+ Growth | Revenue)” and “January | Expenses *(1+ Growth | Expenses)” which are clearly distinct narratives.



	Growth	January	February
Revenue	10%	100.00	110.00
Expenses	5%	60.00	63.00
Operating profit		40.00	47.00

Figure 4.9: An illustration of the SUMWISE spreadsheet extension.

Numbers, a spreadsheet application developed by Apple Inc. (2013), automatically creates named ranges over given data after adding data and headers. For example, in Figure 4.10, the named ranges, “cars sold” and “total income” were used to create a formula for “average price” as “total income / cars sold” that has populated column D and the same formula can be used for the entire column as the row number is not required. Our tool differs from the named range feature in Numbers in three ways. First, in our tool we use the “column name | row name” to come up with a symbolic name for a cell while in Numbers, the symbolic name is derived only from the column name or header. We feel this can bring confusion as to which row is being referred to for a particular reference cell. So for example, in Figure 4.10, we could have “total income | January / cars sold | January” to represent “total income for the month of January divided by the number of cars sold in the month of January to get the average price of a car sold in the month of January”. This would distinguish data for each month. Secondly, in our tool, we do not replace the spreadsheet-specific formula in the formula with a domain narrative as is the case in

the Numbers spreadsheet application. We do this to ensure that we do not create a very different spreadsheet environment that is so much different from what users are used to. For example, in Figure 4.1, a formula given in domain terms as “SUM(Jan | James Bourne ... Jan | Jasmine Hunt)” has its corresponding spreadsheet specific formula still displayed as “SUM(C5:C8)” in the formula bar and we also clearly highlight all the referenced cells of an active formula cell. Thirdly, our tool also differs from the Numbers named range feature in the spatial location and display of the domain narratives on the spreadsheet. In our tool, the domain formula is displayed as a narrative in a box just right below the active formula cell and not in the formula bar as in the Numbers spreadsheet application.

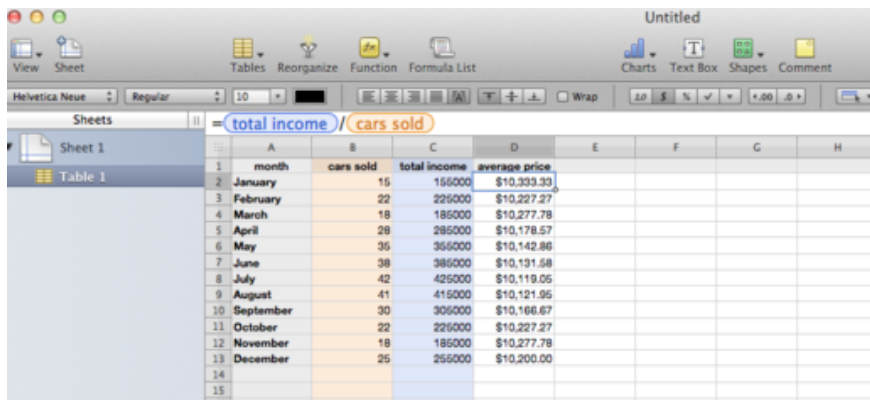


Figure 4.10: An illustration of named ranges in Numbers, the spreadsheet application.

Spreadsheet Auditor for Excel by JabSoft Ltd. (2013) is one other commercial product that uses symbolic names and formula translation similar to what we have in our tool though with different notation and spatial location of the domain narratives.

Chadwick et al. (2001) also translated spreadsheet formulae by replacing cell references with symbolic names based on cell labels within a spreadsheet. This technique was evaluated through a survey of 63 students (non-professionals) on their preference rankings in which the usual notation of Microsoft Excel with cell references was better accepted than when cell references were replaced with

labels. We, however, put forward that the study by Chadwick et al. (2001) is problematic because their study participants were non-professionals and the study task was just a preference ranking exercise rather than a spreadsheet task.

Abraham and Erwig (2004) also extract higher level humanized formulae for spreadsheets based on labels in spreadsheets. Labels are referred to as “headers” while what we call “domain narratives” they refer to as “units”. In addition to differing syntactical notation from our work, they use the inferred units for carrying out automatic consistency checking of formulae, whereas in our work we show the domain narratives directly to the user to help them understand the formula calculations.

Model-driven spreadsheet development approaches such as ClassSheet models (Engels & Erwig, 2005; Cunha et al., 2010) also translate spreadsheet formulae to more humanized higher level object oriented style formulae. Our approach, however, differs from model-driven spreadsheet development approaches such as ClassSheet models in several ways.

Much as ClassSheet models also present formulae at a ‘higher level’, they anecdotally assume that higher level object oriented formulae are useful. In our work, we do not assume this, and our proposed solution is based purely on a theory of mental models perspective which does not necessarily correspond to model-driven development of spreadsheets.

Furthermore, in our approach, we took effort to make sure that the tool should not be completely different from the traditional spreadsheet environment since people are used to develop spreadsheets in an ad-hoc, interactive and incremental manner. This is not the case with model-driven approaches.

Our tool also automatically highlights referenced cells of an active formula cell. On the other hand, in many spreadsheet systems, such as Microsoft Excel, referenced cells are highlighted in edit mode only.

4.7 CONCLUSION

In this chapter, we reported a domain terms spreadsheet visualization tool that we developed to demonstrate that it is possible to have an easy-to-use spreadsheet understanding and debugging tool that relieves users from spreadsheet details and lets them utilize more of their mental model of the application domain. We gave a description of the tool as well as potential uses of the tool. This was to partly answer research question **RQ4** – *Is it possible to develop a spreadsheet understanding and debugging tool that relieves users from spreadsheet details and lets them utilize more of their mental model of the application domain?* We also compared our tool with similar tools and techniques. It is important, however, to note that all the above stated tools and techniques anecdotally assume that symbolic names and formula translation can be useful to users nor has there been psychological justification of the usefulness of symbolic names and formula translation. An empirical evaluation of the usefulness of symbolic names and formula translation in the spreadsheet process such as in debugging should therefore be in order. We present an empirical evaluation of the tool in the next chapter to completely answer the aforementioned research question.

Bennett Freinderson Kankuzi: Deficiencies in Spreadsheets - A Mental
Model Perspective

5 Evaluation of the Spreadsheet Visualization Tool⁵

5.1 INTRODUCTION

In the previous chapter, we reported a domain terms spreadsheet visualization tool that we developed to aid in spreadsheet comprehension and debugging. In this chapter, we give an evaluation of the usability of the tool, particularly when locating errors in a spreadsheet. In particular, we keep the following research questions in mind:

RQ5 – *Can the tool be learned easily by users?*

RQ6 – *Can the tool help users in debugging their spreadsheets?*

RQ7 – *Are users satisfied with the tool?*

We also still keep in mind the following research question which was partly answered in the previous chapter (Chapter 4):

RQ4 – *Is it possible to develop a spreadsheet understanding and debugging tool that relieves users from spreadsheet details and lets them utilize more of their mental model of the application domain?*

We believe that the empirical evaluation of our tool in locating errors in spreadsheets could also be one contribution to determine the usefulness of the special tools mentioned in Chapter 4. We also believe that the empirical evaluation of our tool could also gauge the usefulness of standard range names since their usefulness has

⁵The condensed version of this chapter is published as a refereed paper: Kankuzi, B., & Sajaniemi, J. (2014). Visualizing the problem domain for spreadsheet users: A mental model perspective. In *Proceedings of the 2014 IEEE Symposium on Visual Languages and Human-Centric Computing* (pp. 157 – 160). Washington, DC, USA.

been questioned (Panko & Ordway, 2005; McKeever, McDaid, & Bishop, 2009).

5.2 EVALUATION OF THE TOOL

In order to evaluate the usability of our tool and to study its effects on spreadsheet authors' mental representations, we conducted an empirical study with frequent spreadsheet users working on a debugging task. We adapted usability attributes by Nielsen (1994), namely learnability, efficiency and satisfaction. Learnability is defined as how easy it is for users to accomplish basic tasks the first time they encounter a tool; efficiency is defined as how quickly users can perform tasks once they have learned a system; and satisfaction is defined as how pleasant it is to use a tool and thus how satisfied one is with the tool.

If people cannot understand what a tool does, then the tool cannot be useful. On the other hand, even if people understand the notation of a tool, the tool can still be useless. Therefore, it is important to study first whether the notations used in a tool are understandable, i.e., whether people can learn to use the tool, and only then to determine whether a tool helps in some tasks so as to complete the answer to the research question RQ4. Therefore, the evaluation was divided into four parts:

- (i) Usability of the tool in terms of its learnability
- (ii) Usability of the tool in terms of its efficiency
- (iii) Usability of the tool in terms of the satisfaction of users
- (iv) The effect of the tool on users' mental models

It is important to note that efficiency and satisfaction usability parameters were only evaluated upon participants demonstrating that they had learned and understood the tool.

5.2.1 Participants

The evaluation involved 12 volunteer participants who were also recruited through word of mouth. The participants were frequent users of spreadsheets. Accountants frequently use spreadsheets in their profession and as such, they were our choice of participants in the evaluation. The accountants are of senior managerial positions as such they are directly involved in authoring of spreadsheets unlike their juniors who just use already developed spreadsheets. All participants use Microsoft Excel for their spreadsheet applications. The participants had also not participated in the previous study (the study reported in Chapter 3). We managed to recruit one woman and eleven men. Each participant in the study undertook study tasks at their place of work.

5.2.2 Learnability

Methodology

The researcher visited each participant at their place of work and demonstrated how the tool works. The demo spreadsheet used was sourced from Duggirala (2012) and is partially depicted in Figure 4.1 in Chapter 4. In the demonstration, the researcher highlighted and translated to spreadsheet terms the following two domain narratives on a printed copy of the demo spreadsheet:

- $SUM(Q1 | Mark Watts \dots Q1 | Gill Smith)$
- $Nov | Total North + Nov | Total South +$
 $Nov | Total Wales + Nov | Total Scotland$

The participant was then required to highlight on a hard copy of the same demo spreadsheet the range of cells being given in as a domain narrative (*the highlighting task*) as well as to write down an equivalent spreadsheet-level notation of the same (*the translation task*). Scores in each task were then recorded. A 50% score in each task was considered to be good enough for the participant to con-

tinue in the study. The following five domain level notations were given:

- SUM(Jan | James Bourne ... Jan | Jasmine Hunt)
- Jan | Simon Campbell + Jan | Glen Wilks
- AVERAGE(Sep | Jane Hill ... Q3 | Jane Hill , Nov | Jane Hill)
- SUM(May | Total North , May | Total South , May | Total Wales , May | Total Scotland)
- SUM(Q1 | Total North , Q1 | Total South , Q1 | Total Wales , Q1 | Total)

Results

For the highlighting task, the mean score was 85% (min 60%, max 100%). The most common error was the leaving out of some cells that were supposed to be highlighted particularly for the expression

SUM(Q1 | Total North , Q1 | Total South , Q1 | Total Wales , Q1 | Total)

The foregoing expression deliberately included “Q1 | Total” which many participants ignored to highlight because it is generally erroneous to include a “Total” to the items that make up the “Total” itself. Another common error in the highlighting task was in the expression

AVERAGE(Sep | Jane Hill ... Q3 | Jane Hill , Nov | Jane Hill)

Many participants seemed not to know that the “AVERAGE” function can take in cells that are not contiguous. In this case, “Nov | Jane Hill” was usually left out.

For the translation task, the mean score was 83% (min 60%, max 100%). A common error was failing to write out the corresponding translated expression to reflect the original expression. For example, instead of translating the expression

AVERAGE(Sep | Jane Hill ... Q3 | Jane Hill , Nov | Jane Hill)

to “AVERAGE(M14:N14, P14)”, some participants wrote “M14:N14, P14”. So, essentially, they could figure out the input cells to the formula but forgot to write down exactly the corresponding required translation. Another common error in the translation was re-writing the same expression in another equivalent form. For example, instead of translating

SUM(May | Total North , May | Total South , May | Total Wales , May | Total Scotland)

to the expected “SUM(H9, H18, H25, H31)” some participants wrote “H9 + H18 + H25 + H31” which is not wrong per se, but does not reflect the corresponding original expression. So, again, they could figure out the input cells to the formula but did not write down exactly the required translation. Errors in the highlighting task were reflected in the translation task. For example one participant missed out a cell in the highlighting task and that very cell was also missed out in the translation.

A box-plot for scores in the two sub-tasks is illustrated in Figure 5.1.

Discussion

Overall the participants performed well on both the highlighting and translation tasks considering the very short introduction presented to them.

Many participants failed to translate an expression correctly despite correctly figuring out input cells to the formula. We put forward that this could be the case, because the participants may find figuring out input cells to a formula to be more important than dwelling much on the syntactical intricacies of a particular expression.

Since we considered a 50% or above score to be good enough for the tasks, we concluded that each participant individually per-

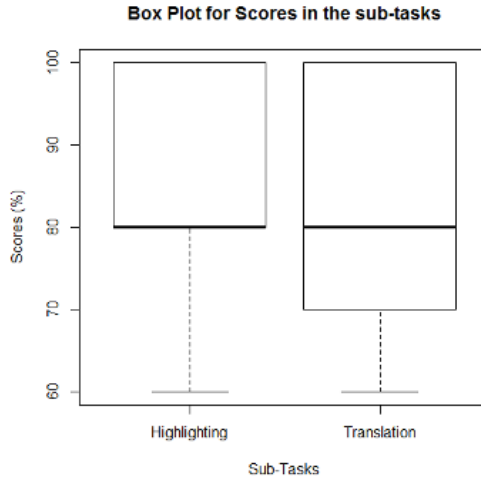


Figure 5.1: A box-plot illustration for scores in the two sub-tasks.

formed reasonably well in the learnability tasks and could proceed to the other evaluation tasks.

5.2.3 Efficiency

Methodology

The efficiency of the tool was evaluated with a within-subjects design experiment where the task was to locate errors in a spreadsheet either without the tool or with the tool. We prepared two roughly equivalent spreadsheets (S1 and S2) with similar seeded errors. The spreadsheets were sourced from the EUSES spreadsheet corpus (Fisher & Rothermel, 2005). Six participants debugged their first spreadsheet without the tool and the second spreadsheet with the tool (three starting with spreadsheet S1 and three starting with spreadsheet S2); the other six debugged their first spreadsheet with the tool and the second without the tool (and again half of them starting with spreadsheet S1).

The errors seeded in the spreadsheets are typical errors spreadsheet users make as adapted from Raffensperger (2008) and

Duggirala (2012). Error types of seeded errors and the number of seeded errors in spreadsheets S1 and S2 are given in Table 5.1.

Each participant was individually requested to locate errors in a given spreadsheet. Times for working on each spreadsheet were also recorded. Participants were at liberty to stop working on the assigned spreadsheet at any time they felt they had found enough errors.

Table 5.1: Error types of seeded errors and their corresponding number of seeded errors in spreadsheet S1 and spreadsheet S2.

Error Type	Errors in Spreadsheet S1	Errors in Spreadsheet S2
(A) <i>Formula with no precedents (hard-coded values)</i>	10	9
(B) <i>Numbers formatted as text</i>	2	1
(C) <i>Formula accidentally overwritten with constants</i>	2	2
(D) <i>Formula missing some range</i>	1	1
(E) <i>Formula incorrectly copied such that it refers to neighbouring range</i>	1	1
(F) <i>A numerical formula pointing to a label when it should point to a number</i>	1	1
(G) <i>A wrong problem domain formula</i>	2	2

Results

We excluded error type A from the analysis of the results. We did this for several reasons: hard-coded values are not necessarily er-

rors and indeed almost all the participants ignored them anyway; most of the error occurrences of this type were not seeded as each of the spreadsheets had them in their original form; and the number of occurrences of this error type is too high as compared to the other types. Error type A therefore deflates the percentage of errors located and hence affecting the true picture of the results. Removing this error type significantly improves the means of located errors from 32% (without tool) and 44% (with tool) to 53.9% (without tool) and 66.6% (with tool).

In view of the above, our analysis therefore involved six error types and nine errors in spreadsheet S1 (excluding the ten errors for spreadsheet S1 from error type A) and eight errors in spreadsheet S2 (excluding the nine errors for spreadsheet S2 from error type A).

Relative frequencies of errors located without the tool and with the tool for each participant are illustrated in Figure 5.2. The mean of the relative frequency of errors located without the tool was 53.9% (S.D. 23.4) and with the tool 66.6% (S.D. 23.9). The relative frequencies of located errors for each error type without the tool and with the tool for all participants are illustrated in Figure 5.4. We ran a two-way repeated measures analysis of variance (ANOVA) with two within-subjects factors with the relative frequency of located errors as the dependent variable and two independent variables: the usage of the tool (without the tool and with the tool) and error type (six different error types). We used an alpha level of 0.05 for all the tests. There were significant main effects of usage of the tool ($F(1, 11) = 7.244, p = 0.021$) and of error type ($F(5, 55) = 4.735, p = 0.001$). However, there was no significant two way interaction of usage of tool and error type ($F(5, 55) = 2, p = 0.093$).

The time used by each participant in debugging each spreadsheet varied from 7 to 28 minutes (mean 15:55) without the tool, and also from 7 to 28 minutes (mean 18:20) with the tool. We calculated *error detection rate* as the average time taken to locate an error illustrated in Figure 5.3. One participant (p7) had an exceptionally poor rate when working without the tool and was excluded as an outlier from the analysis. In fact, this participant (p7) started with

Evaluation of the Spreadsheet Visualization Tool

the tool and found 4 errors in 23 minutes (took 5.8 minutes to locate an error) and without the tool he found 2 errors in 21 minutes (took 10.5 minutes to locate an error). And looking at the participant's subjective opinion of the tool, he wrote that he found the tool "good for fast and accurate work". We would thus suspect that the participant after being exposed to the tool might have lost confidence in working without the tool hence spending unusually long times in locating an error. With the exclusion of this participant, the mean was thus 238 seconds (S.D. 151.0) without the tool and 218 seconds (S.D. 125.0) with the tool. A one-way within-subjects analysis of variance (ANOVA) with the error detection rate as the dependent variable and the independent variable being the usage of the tool (without the tool and with the tool) revealed no significant difference ($F(1, 10) = 0.2592, p = 0.622$).

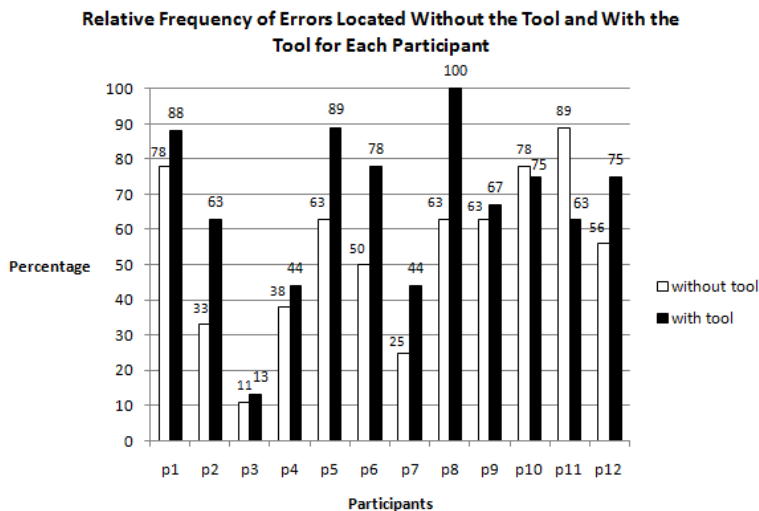


Figure 5.2: Bar chart illustrating relative frequencies of errors located without the tool and with the tool for each participant.

Discussion

There was significant difference in the relative frequency of errors located without the tool (53.9%) and with the tool (66.6%). We thus

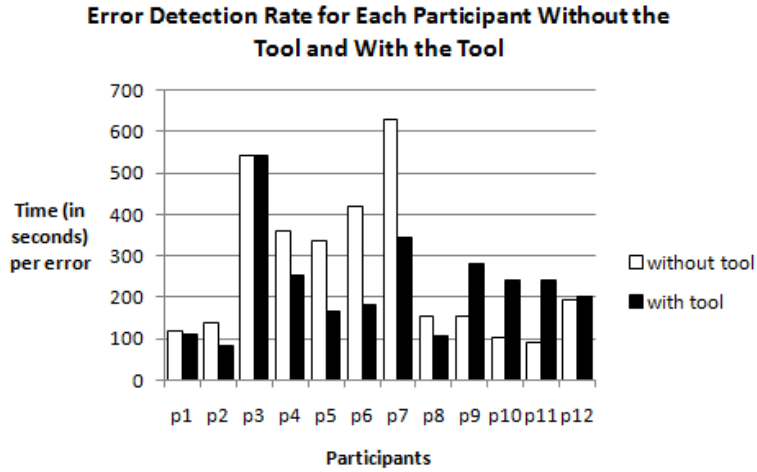


Figure 5.3: Bar chart illustrating error detection rates for each participant without the tool and with the tool.

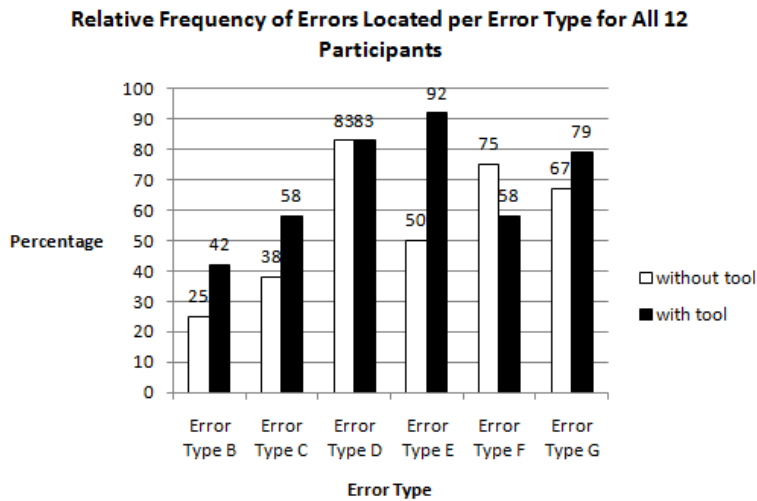


Figure 5.4: Bar chart illustrating relative frequencies of located errors for each error type without the tool and with the tool.

conclude that the tool generally helped participants to locate more errors. In particular, in Figure 5.2, with the tool, participant number eight managed to locate all errors in a given spreadsheet. However, the tool was not helpful to participant number eleven. Participant number three found the least number of errors without the tool and also with the tool. Ironically, this participant had to take considerable time to locate an error without the tool and with the tool (see Figure 5.3). We suspect that maybe the participant found the spreadsheets so unfamiliar as compared to the other participants such that he had to take considerable time to familiarize himself with the spreadsheets given to him. Once again, it is important to note that each participant was at liberty to stop working on the assigned task at any time they felt they had found enough errors in the given spreadsheets and as such the researcher could not intervene in any of the tasks to give leads to a participant to locate more errors in a spreadsheet.

On the other hand, even though errors were found faster with the tool (mean 218 seconds vs. 238 seconds without the tool), the difference was not statistically significant and one might be tempted to argue that the tool does not help in the error detection process. However, there was a strong practice effect as the participants detected errors faster when working with their second spreadsheet ($F(1, 10) = 10.544, p = 0.009$), which weakens the statistical significance of the tool effect on the error detection rate. Thus the tool seems to speed up the process of checking formulae only slightly but makes it clearly more accurate.

Our result differs from that of McKeever et al. (2009) who found that the use of named ranges may lead to a reduction in debugging performance. Their study was, however, quite different from ours. First, the ecological validity of our study was better: participants in the McKeever et al. (2009) study were novices and did not necessarily know the problem domain whereas our participants were experienced spreadsheet users and familiar with the problem domain. Second, named ranges are more error-prone than domain narratives used in our tool, because the domain narratives are generated

automatically by the tool and they are accompanied by a clear visualization of the referenced cells. Thus, our tool provides more opportunities for a user to detect erroneous cell references. We also note that some researchers argue that standard range names can sometimes bring confusion in the spreadsheet understanding process. For example, Panko and Ordway (2005) argue that in selecting ranges for range names, pointing errors can assign the wrong range to a range name and thus although the range will be wrong, but it will appear to be correct in formulae that merely reference the range names. In our case, the domain narratives are automatically generated and each narrative corresponds to the referenced range area and as such a “correct range name but an incorrect range” type of error (Panko & Ordway, 2005) is impossible in our tool.

Based on Figure 5.4, it seems the tool helps users to locate more errors of the following error types:

- Error type E (*formula incorrectly copied such that it refers to neighbouring range, 50% vs. 92%*): The difference between the number of located errors using the tool and without using the tool for this error type was +42 percentage points. We put forward that the tool is providing leverage in this case for two reasons. First, the tool automatically highlights the formula range of an active formula cell and therefore any mismatch with the expected range is quickly taken as a cue for a possible error. Without the tool, a user has no easily available visual cue of the referred area and makes the check using imagery which is more error-prone than a simple visual check. Second, the tool also produces a narrative for each formula and hence if a narrative mismatches with the expected narration, that can be taken as a cue for a possible error.
- Error type C (*formula accidentally overwritten with constants, 38% vs. 58%*): The difference between the number of located errors using the tool and without using the tool for this error type was +20 percentage points. We put forward that the lack of a cue for a presence of a formula or the lack of a narrative

though expected, raises a flag for a possibility of an error.

- Error type **B** (*numbers formatted as text, 25% vs. 42%*): The difference between the number of located errors using the tool and without using the tool for this error type was +17 percentage points. We put forward that this is the case because the narrative captures a number as a row or column header raising a cue for a possibility of an error since numbers are not normally expected to be headers.
- Error type **G** (*a wrong problem domain formula, 67% vs. 79%*): The difference between the number of located errors using the tool and without using the tool for this error type was +12 percentage points. We put forward that this is the case because a domain narrative that does not match with what is expected in the problem domain provides a cue for an error. However, since the difference between the number of located errors using the tool and without using the tool for this error type seems small, we put forward that in general, wrong problem domain formulae easily contradict with what is expected in the application domain hence is easily identified as an error.

There was no difference between using the tool and not using the tool for **error type D** (*formula missing some range, 83% vs. 83%*). We put forward that this might be the case because whether using a tool or not, one debugging technique for many spreadsheet users is to look at whether some cells in the expected range of a formula are missing.

Without using the tool, participants performed better in locating errors in **error type F** (*a numerical formula pointing to a label when it should point to a number, 75% vs. 58%*). We suspect that the inclusion of labels in a range of a formula easily contradicts with the expected range of a formula hence even without a tool, spreadsheet users easily catch this type of an error.

We therefore put forward that the tool is providing leverage for several reasons. First, the tool produces a domain narrative for

each formula and if a narrative does not match with the expected narration, that can be taken as a logical cue for a possible error (error types D, E, and G). Second, the tool automatically highlights the formula range of an active formula cell and therefore any mismatch with the expected range can be taken as a visual cue for a possible error (error types D and E). Third, the tool highlights all formula cells with a magenta right border and therefore the lack of a cue for a presence of a formula raises a flag for a possibility of an error (error type C). Fourth, narratives capture only textual cells as row or column headers raising a cue for a possibility of an error if the contents of a numeric data cell is included in a narrative (error type B). The results indicate that the tool generally helps users to catch more errors in spreadsheets although different aspects of the tool may be more helpful for some error types than others.

5.2.4 Satisfaction

After finishing the task of locating errors, participants were simply requested to write down their opinion of the two scenarios in terms of how they find it easier to locate errors as well as any suggested improvements to the tool.

Eleven out of the twelve participants found the tool helpful in locating errors. For example, one participant wrote: *“Situation A [with tool], was easier than B [without tool]. Because it has some finding tips – there are the narrations in the formula cells, and also the highlights but also the magenta colours against the [formula] cells.”* Another participant also wrote: *“The inclusion of higher level formulae assisted quite a lot to check whether indeed those areas are captured.”* Another participant also wrote: *“It was easier with the helper because you could easily see the affected cells before doing anything else ... Perhaps it would make work easier if the helper was embedded into the system to enable even those who are not well conversant with Excel to see and make easier interpretation.”*

On the other hand, one participant said that he found the tool confusing as he is used to the “normal Excel.” Ironically, this participant found more errors with the tool than when locating errors

on a spreadsheet without the tool. This demonstrates that “benefiting” from a tool is not the same as “liking” a tool. However, overall, the subjective opinion of the participants on the tool shows that, generally, participants found the tool useful.

5.2.5 Effect on mental models

Methodology

Nine of the twelve participants wrote down explanations for each of the located errors in the assigned tasks. The participants were not asked to write the descriptions, but they wrote them voluntarily. We therefore took advantage of this to analyze the effect of the tool on the mental models of participants based on the characterization of spreadsheet authors’ mental models that we found in the study in Chapter 3, i.e., to what extent they deal with the application in domain terms, real-world terms, and spreadsheet terms.

It is important to note that the participants’ explanations were going beyond the contents displayed in the narrative boxes. For example, for some cell, the visualization produced a narrative “8112 | TOTAL FUND BALANCES” which does not make sense and the participant indeed correctly identifies the error in his explanation as “Total liabilities and fund balances. There is an error of not adding the total liabilities to the total figure.” So, the term “Total liabilities” is not mentioned in the narrative, but represents the participant’s understanding of the error. Thus, participants’ explanations can be said to represent their mental models in the error locating task.

Each of the explanations was analyzed and classified using the same technique we used in the study reported in Chapter 3. We adapted the inter-rater reliability verified program summary analysis technique by Good (1999) in which each object/noun is classified as spreadsheet specific or domain-specific or real-world. The categories are not exclusive, i.e., an object can be, say, both domain-specific and real-world.

Results

The relative frequencies of the three object reference types in error descriptions, without the tool and with the tool, are given in Table 5.2 and illustrated in Figure 5.5.

Table 5.2: Relative frequencies (%) of different object reference types in error descriptions without the tool and with the tool.

		Condition				Total	
		Without tool		With tool		M S.D.	
		M	S.D.	M	S.D.		
Obj.ref. type	Real-world	11.1	9.9	16.3	10.2	13.7	10.1
	Domain	23.3	19.8	38.2	18.1	30.8	19.9
	Spread-sheet	92.8	11.1	73.2	12.5	83.0	15.3

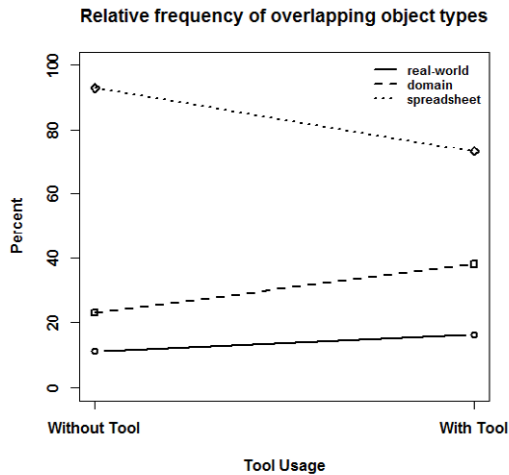


Figure 5.5: A line illustration of relative frequencies of overlapping object reference types in error descriptions.

We ran a two way repeated-measures analysis of variance (ANOVA) with two within-subjects factors with the relative fre-

quency of each object type as the dependent variable and two independent variables: tool usage with two levels (without tool and with tool) and type of object reference (real-world, domain specific and spreadsheet specific). We used an alpha level of 0.05 for all tests. There was no main effect of tool usage ($F(1,8) = 0.005$, $p = 0.945$). However, there was main effect of type of object reference ($F(2,16) = 71.48$, $p < 0.0001$). There was also a significant interaction of tool usage and type of object reference ($F(2,16) = 19.12$, $p < 0.0001$). Looking at the means as also illustrated in Figure 5.5, tool usage is a very influential factor as it impacts differently on the object references in each of the two scenarios particularly for domain references and spreadsheet references. However, it is also clear that tool usage does not affect real-world object references.

The foregoing analysis was on data whose coding scheme allowed that certain terms to be overlapped in the classification, e.g., “zeroes” coded as “yes” for both categories. We therefore did further analysis of the data with “pure” categorization where such terms are left out, i.e., looking at object references that belong to a single mental model only. Since the tool usage does not affect real-world object references, we did not include “pure real-world object references” in this analysis resulting in two opposite categories: “pure spreadsheet specific object references” and “pure domain specific object references”.

Without the tool, the mean of the relative frequencies was 90.8% for pure spreadsheet object references and 9.2% for pure domain object references. With the tool, the corresponding figures were 71.1% and 28.9%. The change affected by the tool is statistically significant ($F(1, 8) = 42.45$, $p = 0.0002$).

Discussion

The use of different types of words is an indication of the mental model a person is using at a moment (Pennington, 1987). The study presented in Chapter 3 found that spreadsheet authors have at least three mental models of a spreadsheet: the real-world model that comprises general knowledge of the world around us and utilized

in the spreadsheet; the domain model that represents knowledge of the problem domain and the functionality of the spreadsheet in problem domain or application terms; and the spreadsheet model that codes the expressions and data relationships in the spreadsheet. That study indicated that authors explain their spreadsheets mainly in terms of real-world and problem domain concepts; in debugging, they constantly switch between problem domain concepts and spreadsheet-specific concepts, although they mainly use spreadsheet-specific concepts to fix an identified error.

In this study, participants used mostly spreadsheet terms when describing an error in the without tool case. This is in line with the study reported in Chapter 3, i.e., with no tool support it was found that the spreadsheet model was also prominent when locating errors.

With the tool, the situation is different: when compared to the without tool case, the prominence of the spreadsheet model now decreases whereas the prominence of the domain model increases. Thus, the tool affected a significant change in the roles of these two models.

On the other hand, the amount of real-world terms was small and not affected by the use of the tool. We put forward that this is the case because when dealing with errors, the real-world model is not as prominent as the two other mental models.

The further analysis with “pure” categorization of data gives further evidence to the effect of the tool in changing the roles of the spreadsheet and domain mental models: while participants deal with errors almost solely in spreadsheet terms without the tool, the tool gives them a more balanced representation that combines the spreadsheet and domain models.

It is important to note that in practice very few errors can be understood in the spreadsheet model, only. Most of the errors in the current study did not belong to the domain model, either. Most often, an error is a combination of domain and spreadsheet models and requires understanding the mapping between these two. The tool made the mapping available to the participants as they

could utilize both models simultaneously. This phenomenon suggests that the tool bridges the gap between the two mental models. The earlier study presented in Chapter 3 demonstrated that when locating and fixing an error, one must constantly switch back and forth between the domain model and spreadsheet model, which requires frequent use of the connection between problem domain concepts and their spreadsheet model counterparts. This study demonstrated that the tool affected a more balanced use of these two models and improved participants performance in spreadsheet debugging.

We therefore put forward that the tool improves the mapping between the spreadsheet and domain models which makes understanding and debugging spreadsheets more efficient.

5.3 THREATS TO VALIDITY OF STUDY

The spreadsheets that the participants used in this study were not created by the participants themselves and as such there could be an effect of unfamiliarity with the spreadsheets on their performance. However, the spreadsheets were quite simple and from the accounting field which the participants themselves belong to. Moreover, spreadsheet users frequently use spreadsheets created by others (Mittermeier & Clermont, 2002). As such, using spreadsheets created by others could help to realistically gauge the efficacy of the tool on its users.

The small number of participants as well as their narrow occupation distribution (professional accountants only) also poses a risk of validity on this research. On the other hand, the findings could be expressed in general terms that are not tied to any profession.

Each participant was at liberty to stop working on the assigned task at any time they felt they had found enough errors in the given spreadsheets. This could have led some participants to find more errors than others. However, this is not a problem as the study was a within-subject design experiment in which each participant was compared to themselves in the two conditions (using the tool

or not) and we could accordingly use repeated measures statistical analyses.

Finally, the participants were not asked to write the descriptions of the errors they located, but they wrote them voluntarily. Therefore the result is not based on an artificial task and hence maintains the ecological validity of the study.

5.4 CONCLUSION

In this chapter, we reported the empirical evaluation of the usability of the domain terms spreadsheet visualization tool that we presented in the previous chapter. We also presented our findings on the effects of the tool on participants' mental models in a debugging task.

The tool was found to be learnable and thus answering research question **RQ5** – *Can the tool be learned easily by users?* The tool also helped the participants to locate more errors in spreadsheets. This finding provides the answer to research question **RQ6** – *Can the tool help users in debugging their spreadsheets?* Participants also found the tool useful in the error locating task and thus were satisfied with the tool. This finding provides the answer to research question **RQ7** – *Are users satisfied with the tool?* The analysis of the effect of the tool on the participants' mental models revealed that the tool makes the prominence of the spreadsheet model to decrease while at the same time increasing the prominence of the domain model in the user's reasoning. Thus the tool relieves users from spreadsheet details and lets them utilize more of their mental model of the application domain. Hence we put forward that the tool improves the mapping between the spreadsheet and domain models which is crucial in understanding and debugging a spreadsheet. This finding completes the answer to the research question **RQ4** – *Is it possible to develop a spreadsheet understanding and debugging tool that relieves users from spreadsheet details and lets them utilize more of their mental model of the application domain?*

The tool gives out spreadsheet formula narratives in domain

terms, but does not currently allow spreadsheet authors to enter or edit formulae using the same notation. Adding such a feature to the tool would further decrease the need for the spreadsheet model and bring formula editing closer to the domain model. We thus propose an extension to the tool to accommodate this paradigm shift whereby a spreadsheet user should be able to debug a spreadsheet in domain terms rather in traditional spreadsheet terms. Moreover, the tool would also ideally allow spreadsheet creation by writing formulae in domain terms. This is addressed in the next chapter (Chapter 6).

Bennett Freinderson Kankuzi: Deficiencies in Spreadsheets - A Mental
Model Perspective

6 A Case for a Paradigm Shift in Spreadsheets

6.1 INTRODUCTION

We inferred from the empirical study reported in Chapter 3 that tools intended to aid in comprehension of a spreadsheet should make prominent real-world and domain concepts and map those concepts easily to spreadsheet-specific details; and a good debugging tool must not only display possible spreadsheet errors in spreadsheet terms (e.g., cell references) but also in domain terms in order to help spreadsheet authors discern errors with a more pronounced mapping between the problem domain and the spreadsheet.

One thing that therefore becomes clear is that allowing spreadsheet authors to express themselves more at a domain level than just at a spreadsheet specific level could reduce the mental barriers that spreadsheet authors can have in mapping with what they think in their head and how to express that in a spreadsheet. This would thus provide a *paradigm shift* - a shift from only allowing spreadsheet authors to express themselves in a spreadsheet mainly in spreadsheet specific terms to being allowed to do the same in higher-level domain terms. In other words, instead of spreadsheet authors expressing themselves mainly in spreadsheet terms, .i.e., the spreadsheet model, as in the current traditional spreadsheet paradigm when creating spreadsheets or indeed debugging spreadsheets, they should be allowed to express themselves in higher level domain terms (i.e. the domain model). Although some spreadsheet systems, such as Microsoft Excel, allow one to define formulae in domain terms, one has to define cell names separately through named ranges and as such it is error prone and also not

easy to define formulae in domain terms. The SUMWISE spreadsheet extension by Miller et al. (2010) employs user-defined row and column names, rather than the usual A1 (or R1C1) references found in traditional spreadsheets. In other words, spreadsheet authors can use domain terms to define and edit spreadsheet formulae in this environment which completely replaces the traditional spreadsheet environment. However, the usability of the SUMWISE extension has not been empirically evaluated. In the paradigm shift we are proposing, spreadsheet authors should *easily* use domain terms while in the traditional spreadsheet environment. Moreover, we should also have an empirical basis for using domain terms in spreadsheets.

This proposed paradigm shift could therefore be applicable in spreadsheet activities such as:

- (i) When creating a spreadsheet, instead of spreadsheet authors defining their spreadsheets in traditional low-level spreadsheet specific terms such as cell references, they could also do so by using high level domain terms.
- (ii) In debugging, when fixing errors in a spreadsheet, the spreadsheet author could fix the errors in higher level domain terms than in traditional low-level spreadsheet specific terms such as cell references.

We thus also give a demonstration of how this paradigm shift could specifically be implemented in spreadsheet debugging and in particular when fixing spreadsheet errors. This demonstration attempts to answer the research question:

RQ8 – *Is it possible to construct a tool that is based on a different paradigm rather than the current traditional spreadsheet paradigm?*

We also do an evaluation of the proposed tool based on the following research question:

RQ9 – *What are the effects of such a tool on spreadsheet authors' mental models?*

6.2 A PARADIGM SHIFT IN SPREADSHEET DEBUGGING

In the traditional spreadsheet paradigm, debugging a formula requires the spreadsheet author to click on a formula cell and read the formula. And the referenced cells are normally highlighted when one is in edit mode. This is true in spreadsheet systems such as Microsoft Excel. The problem that arises is that this formula is in low-level spreadsheet specific terms and yet the spreadsheet author mainly thinks of the problem being solved on a spreadsheet in higher level domain terms. We found in the study in Chapter 3 that when explaining a spreadsheet, spreadsheet authors mainly do so in domain/real-world terms and therefore it would be appropriate that even in other activities such as debugging, spreadsheet authors should also be allowed to think in domain/real-world terms. And the study in Chapter 5 showed that with formulae expressed in higher level domain terms, participants were able to locate more errors than in the traditional spreadsheet paradigm. With this in mind, it would therefore be appropriate that the spreadsheet author also be given another view of the formula in domain terms and they should also be allowed to easily fix errors in domain specific terms that could then be reflected in low level spreadsheet specific terms.

We demonstrate a paradigm shift in spreadsheet debugging through a prototype tool that we have developed that allows one to be able to fix errors in a spreadsheet using higher level domain terms. The tool is an extension of the spreadsheet visualization tool that we presented in Chapter 4. We call this prototype tool a paradigm shift spreadsheet visualization tool or paradigm shift tool in short. The paradigm shift tool extends the tool defined in the previous chapter by providing the ability to allow one to fix errors in higher level domain terms. The tool thus displays formulae in domain terms but also allows one to fix errors using domain

terms. Currently, the tool does not allow entering the formulae in domain terms at the very beginning as it is just a prototype tool. We deliberately excluded the creation of formulae from scratch in domain terms as the purpose of the tool is to demonstrate proof of concept of the paradigm shift we are proposing.

The tool has the following features and characteristics:

- (i) The tool displays formulae in “more user friendly” domain terms and also allows one to fix errors in higher level domain terms.
- (ii) Cells that are being referred to in the domain terms narrative are also automatically highlighted and their background colour matches with the background colour of the narrative. In many spreadsheet systems, such as Microsoft Excel, referenced cells are highlighted in edit mode. We have thus applied the Gestalt Law of Similarity whereby objects of the same colour are naturally perceived as related (Wertheimer, 1938). In this case, matching background colours of the narrative and highlighted cells helps the user to automatically perceive the two as related.
- (iii) All formula cells are marked with a magenta right border so that a user can have a general overview of all formula cells and thus could actually help one to differentiate formula cells from text cells as well as input number cells.
- (iv) Each domain terms narrative is shown when a formula cell is active and the narrative is positioned a little bit lower than the active formula cell to avoid distractions when navigating through the cells.
- (v) To track changes automatically, the domain terms narratives are automatically re-generated as one works through the spreadsheet hence providing real-time interactivity. Automatic generations of visualizations is important because users are

not comfortable with tools that require much user intervention

(Sajaniemi, 2000).

- (vi) The tool is superimposed on the spreadsheet display. This is important because users may find it tedious and confusing to determine the correspondence between a separate visualization and the spreadsheet itself (Sajaniemi, 2000).

As already alluded to above, this paradigm shift tool differs from the domain terms visualization tool presented in Chapter 4 in that the domain terms visualization tool just displays formulae in domain terms but does not also allow one to fix errors in higher level domain terms as the paradigm shift tool does.

6.2.1 Implementation

The tool is implemented as an add-on to a popular spreadsheet system, Microsoft Excel, so that it should not be completely different from the traditional spreadsheet environment. Again, this was to take advantage of existing user experience as their familiarity with existing tools and techniques could become worthless if the tool is too different from what they know and how they are used to work (Kulesz, 2011).

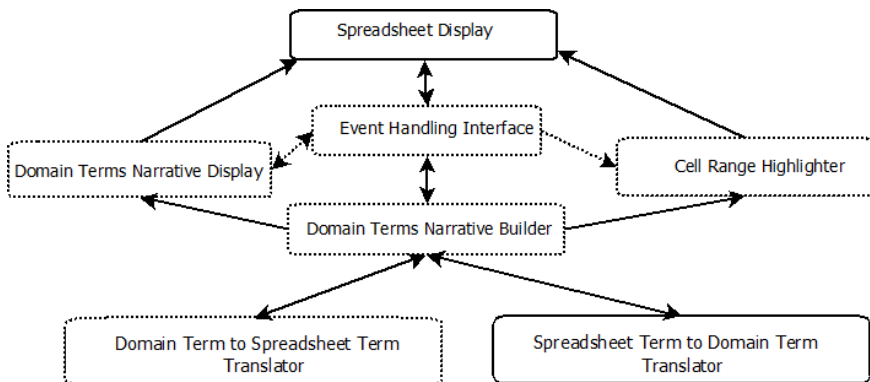


Figure 6.1: An illustration of a high-level architecture of the paradigm shift tool.

Figure 6.1 depicts a high-level architecture of the paradigm shift tool. In the tool, events such as editing a domain terms narrative (a higher level formula) trigger the domain terms narrative builder to translate the domain terms to spreadsheet terms using the “domain term to spreadsheet term” translator and thereafter the affected cells are highlighted accordingly.

The tool thus has the following components: Domain Terms Narrative Display (already described in Chapter 4); Domain Terms Narrative Builder (already described in Chapter 4); Domain Term to Spreadsheet Term Translator – translates a particular domain term to its corresponding low-level spreadsheet specific term; Cell Highlighter (already described in Chapter 4); and Event Handling Interface – captures events such as editing of a domain narrative.

6.2.2 The paradigm shift tool in action

We demonstrate the tool in action in the “Sales Report” spreadsheet depicted in Figure 6.2 up to Figure 6.4. In Figure 6.2, cell C18 has an error – the January Totals for the South region is missing out the sales by Mark Watts. In spreadsheet specific terms, the formula, SUM(C14:C17), in cell C18, is missing out cell C13. In domain terms, the corresponding narrative given is “SUM(Jan | Jane Hill ... Jan | Gill Smith)”. To correct this error, the spreadsheet author has two options: to either fix it by correcting the spreadsheet specific formula by changing it to SUM(C14:C17) or by editing the domain terms narrative from “SUM(Jan | Jane Hill ... Jan | Gill Smith)” to “SUM(Jan | Mark Watts ... Jan | Gill Smith)”. The second option is closer to what a spreadsheet author thinks in their head because the error is that “sales by Mark Watts” have been left out. Therefore replacing “Jane Hill” with “Mark Watts” should correct the problem. And indeed, using our tool, we demonstrate a spreadsheet author editing the domain narrative appropriately as in Figure 6.3. The corrected domain terms narrative is then exited to effect the change as in Figure 6.4. The tool user can exit the domain terms narrative by clicking in any other cell. The effected change can be verified by clicking on the corrected formula cell such as in Figure 6.5 and

A Case for a Paradigm Shift in Spreadsheets

indeed the domain narrative is reflecting the change and the highlighted cells provide a further confirmation of the change. A check on the spreadsheet specific formula on the formula bar also reflects the change – SUM(C14:C17) is now SUM(C13:C17).

SALES REPORT		Jan	Feb	Mar	Q1	Apr	May	Jun	Q2	Jul	Aug	Sep	
Region: North													
James Bourne		4,521	2,863	3,802	11,925	5,698	4,137	3,627	13,462	5,100	2,280	4,207	
Chris Hewitt		3,510	4,882	3,915	12,056	3,843	1,413	4,187	9,443	2,021	1,205	2,468	
Pat Hill		5,213	4,557	2,187	10,562	1,070	4,424	1,240	6,734	4,128	2,331	3,849	
Jasmine Hunt		4,175	2,681	2,257	9,112	3,588	4,666	3,666	11,920	5,410	2,719	1,934	
Total North		17,419	14,983	12,161	43,655	14,199	14,640	12,720	41,559	16,659	16,659	12,458	
Region: South													
Mark Watts		3,400	5,329	4,734	13,463	1,450	4,191	3,109	8,750	2,097	2,354	1,837	
Jane Hill		3,608	4,922	4,711	13,241	1,463	1,204	2,342	5,009	3,402	5,269	3,625	
Roger West		2,359	5,273	5,438	13,070	2,280	4,882	3,915	11,077	5,498	2,806	3,061	
Gill Smith		4,487	2,638	5,100	12,225	1,205	4,557	2,187	7,949	4,043	2,252	5,076	
Total South		12,757	19,415	23,634	59,206	10,993	17,697	15,355	44,045	16,518	18,091	13,599	
Region: Wales													
Gareth Jones		1,063							57	9,113	1,070	1,722	2,176
Tim Stevens		4,424	4,432	4,128	8,876	3,400	3,329	4,734	13,463	3,588	1,732	4,534	
Mike Payne		4,666	1,240	5,410	5,906	3,608	4,922	4,711	13,241	1,450	3,910	2,705	
Total Wales		9,090	5,692		14,782	7,008	10,251	9,445	26,704	5,038	5,642	7,239	

Figure 6.2: An illustration of the paradigm shift tool – step 1 (an error in cell C18 whereby “Mark Watts” has been left out).

Normally, in a spreadsheet, a label can have multiple occurrences. This can be problematic to know which label is the one being referred in the domain narrative. In our tool, we have devised a way to handle this. A user is asked to pick the correct occurrence of a label. The label that can be chosen at a given time is highlighted in colour - yellow if the label is referred to in a “row-wise” context and green if referred to in a “column-wise” context. We illustrate how we deal with the multiple occurrence label problem in the error fixing process illustrated in Figure 6.6 to Figure 6.10. Particularly, in Figure 6.8, since there are two labels titled “Total” in cell E4 and cell A9, the first “Total” label is encountered and the user is asked if that is the correct “Total” label that should be used in the domain narrative particularly for “Production | Total”. This is the wrong label being referred to and as such the user answers in the negative to the question. The “Total” label is highlighted in yellow as it is being referred to in the “row-wise” context in “Production | Total”. If it were being referred to in the “column-wise”

Bennett Freinderson Kankuzi: Deficiencies in Spreadsheets - A Mental Model Perspective

Comment 78

A	B	C	D	E	F	G	H	I	J	K	L	M	
1	SALES REPORT												
2		Jan	Feb	Mar	Q1	Apr	May	Jun	Q2	Jul	Aug	Sep	Q
3													
4	Region: North												
5	James Bourne	4,521	2,863	3,802	11,925	5,698	4,137	3,627	13,462	5,100	2,280	4,207	
6	Chris Hewitt	3,510	4,882	3,915	12,056	3,843	1,413	4,187	9,443	2,021	1,205	2,468	
7	Pat Hill	5,213	4,557	2,187	10,562	1,070	4,424	1,240	6,734	4,128	2,331	3,849	
8	Jasmine Hunt	4,175	2,681	2,257	9,112	3,588	4,666	3,666	11,920	5,410	2,719	1,934	
9	Total North	17,419	14,983	12,161	43,655	14,199	14,640	12,720	41,559	16,659	16,659	12,458	
10													
11													
12	Region: South												
13	Mark Watts	3,400	5,329	4,734	13,463	1,450	4,191	3,109	8,750	2,097	2,354	1,837	
14	Jane Hill	3,608	4,922	4,711	13,241	1,463	1,204	2,342	5,009	3,402	5,269	3,625	
15	Roger West	2,359	5,273	5,438	13,070	2,280	4,882	3,915	11,077	5,498	2,806	3,061	
16	Gill Smith	4,487	2,638	5,100	12,225	1,205	4,557	2,187	7,949	4,043	2,252	5,076	
17	Total South	12,757	19,415	23,634	59,206	10,993	17,697	15,355	44,045	16,518	18,091	13,599	
18													
19													
20													
21	Region: Wales												
22	Gareth Jones	1,063							9,113	1,070	1,722	2,176	
23	Tim Stevens	4,424							13,463	3,588	1,732	4,534	
24	Mike Payne	4,666	1,240	5,410	5,906	3,608	4,922	4,711	13,241	1,450	3,910	2,705	
25	Total Wales	9,090	5,692	-	14,782	7,008	10,251	9,445	26,704	5,038	5,642	7,239	
26													

SUM(Jan | Mark Watts ... Jan | Gill Smith)

Figure 6.3: An illustration of the paradigm shift tool – step 2 (incorrect domain term “Jane Hill” replaced with correct domain term “Mark Watts”).

119

A	B	C	D	E	F	G	H	I	J	K	L	M	
1	SALES REPORT												
2		Jan	Feb	Mar	Q1	Apr	May	Jun	Q2	Jul	Aug	Sep	
3													
4	Region: North												
5	James Bourne	4,521	2,863	3,802	11,925	5,698	4,137	3,627	13,462	5,100	2,280	4,207	
6	Chris Hewitt	3,510	4,882	3,915	12,056	3,843	1,413	4,187	9,443	2,021	1,205	2,468	
7	Pat Hill	5,213	4,557	2,187	10,562	1,070	4,424	1,240	6,734	4,128	2,331	3,849	
8	Jasmine Hunt	4,175	2,681	2,257	9,112	3,588	4,666	3,666	11,920	5,410	2,719	1,934	
9	Total North	17,419	14,983	12,161	43,655	14,199	14,640	12,720	41,559	16,659	16,659	12,458	
10													
11													
12	Region: South												
13	Mark Watts	3,400	5,329	4,734	13,463	1,450	4,191	3,109	8,750	2,097	2,354	1,837	
14	Jane Hill	3,608	4,922	4,711	13,241	1,463	1,204	2,342	5,009	3,402	5,269	3,625	
15	Roger West	2,359	5,273	5,438	13,070	2,280	4,882	3,915	11,077	5,498	2,806	3,061	
16	Gill Smith	4,487	2,638	5,100	12,225	1,205	4,557	2,187	7,949	4,043	2,252	5,076	
17	Total South	16,157	19,415	23,634	59,206	10,993	17,697	15,355	44,045	16,518	18,091	13,599	
18													
19													
20													
21	Region: Wales												
22	Gareth Jones	1,063	3,362	2,021	4,425	4,175	2,681	2,257	9,113	1,070	1,722	2,176	
23	Tim Stevens	4,424	4,452	4,128	8,876	3,400	5,329	4,734	13,463	3,588	1,732	4,534	
24	Mike Payne	4,666	1,240	5,410	5,906	3,608	4,922	4,711	13,241	1,450	3,910	2,705	
25	Total Wales	9,090	5,692	-	14,782	7,008	10,251	9,445	26,704	5,038	5,642	7,239	
26													

Figure 6.4: An illustration of the paradigm shift tool – step 3 (corrected domain terms narrative exited to effect the change).

A Case for a Paradigm Shift in Spreadsheets

SALES REPORT												
	Jan	Feb	Mar	Q1	Apr	May	Jun	Q2	Jul	Aug	Sep	
Region: North												
James Bourne	4,521	2,863	3,802	11,925	5,698	4,137	3,627	13,462	5,100	2,280	4,207	
Chris Hewitt	3,510	4,882	3,915	12,056	3,843	1,413	4,187	9,443	2,021	1,205	2,468	
Pat Hill	5,213	4,557	2,187	10,562	1,070	4,424	1,240	6,734	4,128	2,331	3,849	
Jasmine Hunt	4,175	2,681	2,237	9,112	3,388	4,666	3,666	11,920	5,410	2,719	1,934	
Total North	17,419	14,983	12,161	43,655	14,199	14,640	12,720	41,559	16,659	16,659	12,458	
Region: South												
Mark Watts	3,400	5,329	4,734	13,463	1,450	4,191	3,109	8,750	2,097	2,354	1,837	
Jane Hill	3,608	4,922	4,711	13,241	1,463	1,204	2,342	5,009	3,402	5,269	3,625	
Roger West	2,359	5,273	5,438	13,070	2,280	4,882	3,915	11,077	5,498	2,806	3,061	
Gill Smith	4,487	2,638	5,100	12,225	1,205	4,557	2,187	7,949	4,043	2,252	5,076	
Total South	16,157	19,415	23,634	59,206	10,993	17,697	15,355	44,045	16,518	18,091	13,599	
Region: Wales												
Gareth Jones	1,063							57	9,113	1,070	1,722	2,176
Tim Stevens	4,424	4,952	4,128	8,876	3,400	3,329	4,734	13,463	3,588	1,732	4,534	
Mike Payne	4,666	1,240	5,410	5,906	3,608	4,922	4,711	13,241	1,450	3,910	2,705	
Total Wales	9,090	5,692	-	14,782	7,008	10,251	9,445	26,704	5,038	5,642	7,239	

Figure 6.5: An illustration of the paradigm shift tool – step 4 (verification of effected error correction).

context as in say “Total | Stationery”, the label would have been highlighted in green. It is important to note that in this case we are trying to locate the right “Total” label for “Production | Total”. This step will be repeated also for “Accounting | Total”. In other words, each occurrence of a label in the domain narrative is treated separately. Overall, it takes quite a good number of steps to correct an error which involves a label which has more than one occurrence in a spreadsheet.

6.2.3 Limitations of the paradigm shift tool

The paradigm shift tool we have developed serves to demonstrate that it is possible for a spreadsheet author to fix errors in a spreadsheet in high level domain terms. However, it has some limitations. A major limitation with the tool is that one cannot use it to define a domain terms formula from “scratch”. As such, it can only be used to edit the automatically generated domain terms formulae with already defined corresponding cell-referenced formulae. Other limitations which mainly stem from the implementation approach we used are as follows:

- (i) The names of labels or symbolic names used in a spreadsheet

should not include the following operators: "+", "-", "*", "/", "(", ")" and "...". This pre-condition has been imposed because these operators have been used in the tool to identify domain terms in the narratives.

- (ii) Normally, in a spreadsheet, a label can have multiple occurrences. In this tool, it is laborious and time consuming to effect a domain narrative change when there are multiple occurrences of a label in a spreadsheet. Ideally, it would have been possible to highlight all alternatives at the same time and let the user simply pick the right one.
- (iii) Any input cell to a formula cell needs to have a column label as well as a row label otherwise we will not be able to translate a domain term to a spreadsheet specific cell reference for input cells that lack these. However, this limitation could be a blessing in disguise as it forces a spreadsheet author to define a column label and row label for each input cell which could be good practice to produce readable and self-documenting spreadsheets.
- (iv) Each label has to occupy only one cell otherwise searching for a label that occupies more than one cell is not plausible with this tool. We also note however that in many spreadsheets, labels may occupy more than one cell but we are putting forward that not allowing a label that occupy more than one cell could also be one one way to encourage spreadsheet authors to produce readable and self-documenting spreadsheets.

Although the above limitations make the use of the tool clumsy, it is not a problem because the tool was created just to test the feasibility of the paradigm shift idea and it is not intended for real use. However, one interface enhancement that would be more desirable for the tool would be to allow spreadsheet authors to edit formulae in the displayed domain narratives by not typing by hand but by just pointing and clicking on required labels. This would improve efficiency and accuracy in formula formation using higher

A Case for a Paradigm Shift in Spreadsheets

level domain terms and also eliminate the multiple occurrence label problem as spreadsheet authors would simply select their desired label by pointing and clicking.

	A	B	C	D	E	F	G	H	I	J	K
1											
2	Departmental	Expenses (in USD)									
3											
4		Production	Sales	Accounting	Total						
5	Transport	4000	2000	1500	6000						
6	Stationery	1000	2000	2000	3000						
7	Printing	500	1000	1500	1500						
8											
9	Total	5500	5000	5000	10500						
10											
11											
12											
13											
14											

Figure 6.6: An illustration of the multiple occurrence label problem – step 1: An error occurs in cell E9 in that the sum is leaving out D9 or in domain terms, it is leaving out Accounting total. The correct domain narrative should be “SUM(Production | Total ... Accounting | Total)”. “Sales” needs to be replaced with “Accounting”.

	A	B	C	D	E	F	G	H	I	J	K
1											
2	Departmental	Expenses (in USD)									
3											
4		Production	Sales	Accounting	Total						
5	Transport	4000	2000	1500	6000						
6	Stationery	1000	2000	2000	3000						
7	Printing	500	1000	1500	1500						
8											
9	Total	5500	5000	5000	10500						
10											
11											
12											
13											
14											

Figure 6.7: An illustration of the multiple occurrence label problem – step 2: The erroneous domain narrative is corrected with “Sales” edited out and replaced with “Accounting”.

6.2.4 Evaluation of the paradigm shift tool

In evaluating the tool, we had the following research question in mind:

RQ9 – *What are the effects of such a tool on spreadsheet authors’ mental models?*

In this evaluation, we did not do our evaluation based on usability attributes by Nielsen (1994) as the paradigm shift tool is a

Bennett Freinderson Kankuzi: Deficiencies in Spreadsheets - A Mental Model Perspective

	A	B	C	D	E	F	G	H	I	J	K	L
1												
2	Departmental	Expenses (in USD)										
3												
4		Production	Sales	Accounting	Total							
5	Transport	4000	2000	1500	6000							
6	Stationery	1000	2000	2000	3000							
7	Printing	500	1000	1500	1500							
8												
9	Total	5500	5000	5000	10500							
10												
11												
12												
13												
14												
15												
16												
17												
18												
19												
20												
21												
22												

Label Occuring Multiple Times

There are multiple occurrences of "Total". Is the highlighted "Total" the one you want? Your formula is: (SUM(Production | Total ... Accounting | Total))

Figure 6.8: An illustration of the multiple occurrence label problem – step 3: The user is taken through a sequence of dialog boxes so that they choose the label that is being referred to in the domain narrative.

	A	B	C	D	E	F	G	H	I	J	K	L
1												
2	Departmental	Expenses (in USD)										
3												
4		Production	Sales	Accounting	Total							
5	Transport	4000	2000	1500	6000							
6	Stationery	1000	2000	2000	3000							
7	Printing	500	1000	1500	1500							
8												
9	Total	5500	5000	5000	10500							
10												
11												
12												
13												
14												
15												
16												
17												
18												
19												
20												
21												
22												

Label Occuring Multiple Times

There are multiple occurrences of "Total". Is the highlighted "Total" the one you want? Your formula is: (SUM(Production | Total ... Accounting | Total))

Figure 6.9: An illustration of the multiple occurrence label problem – step 4: This is the right label being referred to and as such the user answers in the affirmative to the question.

A Case for a Paradigm Shift in Spreadsheets

	A	B	C	D	E	F	G	H	I	J	K
1											
2	Departmental Expenses (in USD)										
3											
4		Production	Sales	Accounting	Total						
5	Transport	4000	2000	1500	6000						
6	Stationery	1000	2000	2000	3000						
7	Printing	500	1000	1500	1500						
8											
9	Total	5500	5000	5000	15500						
10											
11											
12											
13											
14											

Formula bar: =SUM(B9:D9)

Callout box: SUM(Production | Total ... Accounting | Total)

Figure 6.10: An illustration of the multiple occurrence label problem – step 5: The correction has been effected and the user can verify this by clicking on the formula cell they were correcting.

prototype rather than a fully functional tool. Instead, we opted to investigate the effect of the tool on mental models in spreadsheet authors. We did this by giving participants a debugging task to do without the tool and with the help of the tool. The debugging task involved an *error description* sub-task and an *error fixing* sub-task.

Methodology

The participants in this study were the same participants that participated in the usability evaluation of the domain terms spreadsheet visualization tool (see Chapter 5) since they were already familiar with the tool except for how to actually use it to fix errors. Therefore the participants were twelve in number. The researcher visited each participant at their place of work. Two spreadsheets were used in the study – the “Group Profit Summary” spreadsheet which we denoted in this study as “S3” and the “Consolidated Statement of Shareholders’ Equity” which we denoted in this study as “S4”. Both spreadsheets were sourced and adapted from the EUSES spreadsheet corpus (Fisher & Rothermel, 2005). These spreadsheets were financial spreadsheets which any accountant could easily understand.

We seeded the two spreadsheets with similar errors as in the usability evaluation of the domain terms visualization tool presented in Chapter 5. The errors were typical errors spreadsheet authors

make as adapted from Raffensperger (2008) and Duggirala (2012). However, we excluded error types A, B and C in this study for the following reasons:

- Error type A (formula with no precedents (hard-coded values)) was excluded because hard-coded values are not necessarily errors although they can have repercussions on the correctness of a spreadsheet since the values are not automatically calculated through formulae.
- Error type B (numbers formatted as text) was excluded because once a number is formatted as text, it is captured in a domain narrative of a formula cell that references it. So, fixing an error of this error type through a domain narrative has to happen not on this cell having a number formatted as text but rather on a formula cell that references it.
- Error type C (formula accidentally overwritten with constants) was excluded because the paradigm shift tool only allows one to edit an already generated domain narrative i.e. a domain narrative cannot be entered from scratch. In this error type, a constant will mean that there is no formula at spreadsheet specific level and there is no domain narrative for that formula hence the formula has to be entered from scratch.

The error types of seeded errors and the number of seeded errors in spreadsheets S3 and spreadsheet S4 are given in Table 6.1.

Six participants were requested to describe and fix the seeded errors in their first spreadsheet without the tool and the second spreadsheet with the paradigm shift tool (three starting with spreadsheet S3 and three starting with spreadsheet S4); the other six were requested to describe and fix the seeded errors in their first spreadsheet with the paradigm shift tool and the second without the tool (and again half of them starting with spreadsheet S3).

A Case for a Paradigm Shift in Spreadsheets

Table 6.1: Error types of seeded errors and their corresponding number of seeded errors in spreadsheet S3 and spreadsheet S4.

Error Type	Errors in Spreadsheet S3	Errors in Spreadsheet S4
(D) <i>Formula missing some range</i>	1	1
(E) <i>Formula incorrectly copied such that it refers to neighbouring range</i>	1	1
(F) <i>A numerical formula pointing to a label when it should point to a number</i>	1	1
(G) <i>A wrong problem domain formula</i>	1	1

Each participant was introduced to the concept of fixing errors in domain terms through a demonstration using a demo spreadsheet just before working on a spreadsheet with the paradigm shift tool. Though both spreadsheets used in the tasks were financial spreadsheets, their structure was different so that the participants could not have a carry-over effect as they moved from the first task to the second task.

In the study task, each participant was informed about each cell containing an error (and this was so because we were not necessarily looking at how many errors they can correctly identify but rather the nature of their mental models as they do the task). The researcher therefore clicked the first erroneous cell in a spreadsheet at hand and let the participant first write down the description of why that cell was erroneous and then fix the error (using the tool if available or without the tool if not available) and also write down how that error was fixed. This was done before moving on to the next cell. This process was repeated for all the errors in the spreadsheet at hand.

The researcher was just clicking on the erroneous cells to avoid influencing the participant's thought process. This way verbal expressions of the location, which could influence how a participant thinks were avoided, and the participants were working with one error at a time. All the errors were given in the same order for all the participants (starting from error type D to G).

For each participant we thus collected what they wrote down in the error description sub-task and error fixing sub-task both when they were not using the paradigm shift tool and when using the tool. Each of the written transcripts was analyzed and classified using the adaptation of inter-rater reliability verified program summary analysis technique by Good (1999) in which each object/noun is classified as spreadsheet specific or domain-specific or real-world as we did in Chapter 3. The categories are not exclusive, i.e., an object can be, say, both domain-specific and real-world.

Results

In the *error description* sub-task, the number of coded object references varied from 5 to 18 (mean 12) without the tool and from 4 to 26 (mean 15) with the tool. The relative frequencies of the three object reference types in the error description sub-task without the tool and with the tool are given in Table 6.2 and illustrated in Figure 6.11.

Table 6.2: Relative frequencies (%) of different object reference types in the error description sub-task without the tool and with the tool.

		Condition				Total	
		Without tool		With tool		M S.D.	
		M	S.D.	M	S.D.		
Obj.ref. type	Real-world	11.2	11.8	11.1	14.2	11.1	12.8
	Domain	13.0	14.7	65.9	20.8	39.6	32.3
	Spread-sheet	91.6	10.2	42.3	21.2	67.0	29.9

A Case for a Paradigm Shift in Spreadsheets

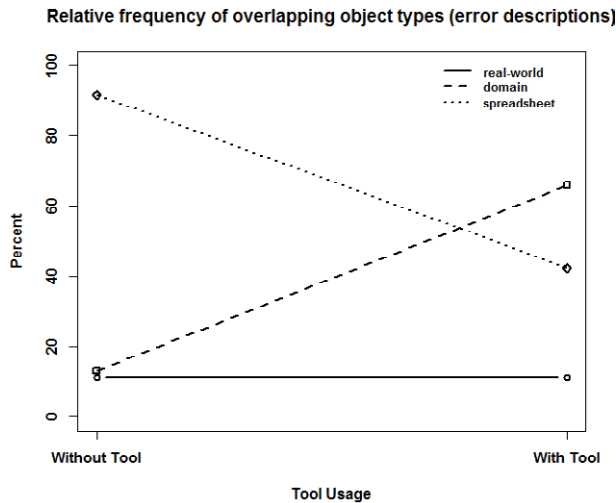


Figure 6.11: A line illustration of relative frequencies of overlapping object reference types in the error description sub-task.

We ran a two way repeated-measures analysis of variance (ANOVA) with two within-subjects factors with the relative frequency of each object type as the dependent variable and two independent variables: tool usage with two levels (without tool and with tool) and type of object reference (real-world, domain specific and spreadsheet specific). We used an alpha level of 0.05 for the tests. There was no main effect of tool usage ($F(1,11) = 0.424$, $p = 0.528$). However, there was main effect of type of object reference ($F(2,22) = 65.22$, $p < 0.0001$). There was also a significant two way interaction of tool usage and type of object reference ($F(2,22) = 40.87$, $p < 0.0001$). Looking at the means as also illustrated in Figure 6.11, tool usage is a very influential factor as it impacts differently in each of the two scenarios particularly for domain references and spreadsheet references. However, it is also clear that tool usage does not affect real-world object references.

In the *error fixing* sub-task, the number of coded object references varied from 4 to 18 (mean 10) without the tool and from 5 to 23 (mean 14) with the tool. The relative frequencies of the three object reference types in the error fixing descriptions without the

tool and with the tool are given in Table 6.3 and illustrated in Figure 6.12.

Table 6.3: Relative frequencies (%) of different object reference types in the error fixing sub-task without the tool and with the tool.

		Condition				Total	
		Without tool		With tool		M S.D.	
		M	S.D.	M	S.D.		
Obj.ref. type	Real-world	4.3	9.1	15.3	20.5	9.8	16.5
	Domain	6.4	14.9	76.4	18.2	40.5	31.9
	Spread-sheet	97.5	8.7	32.5	18.3	65.0	36.1

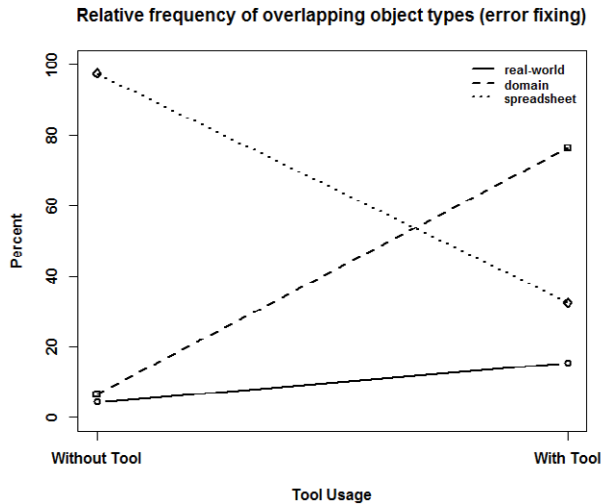


Figure 6.12: A line illustration of relative frequencies of overlapping object reference types in the error fixing sub-task.

We also ran a two way repeated-measures analysis of variance (ANOVA) with two within-subjects factors with the relative frequency of each object type as the dependent variable and two independent variables: tool usage with two levels (without tool and

with tool) and type of object reference (real-world, domain specific and spreadsheet specific). We also used an alpha level of 0.05 for the tests. There was no main effect of tool usage ($F(1,11) = 3.555$, $p = 0.086$). However, there was main effect of type of object reference ($F(2,22) = 75.36$, $p < 0.0001$). There was also a significant two way interaction of tool usage and type of object reference ($F(2,22) = 80.47$, $p < 0.0001$). Looking at the means as also illustrated in Figure 6.12, tool usage is a very influential factor as it impacts differently in each of the two situations particularly for domain references and spreadsheet references. However, it is not clear on the effect of tool usage on real-world object references. A posthoc analysis with a Bonferroni adjusted alpha level, 0.017 (i.e. $0.05/3$), indicated that there is no significant effect of the use of the tool on real-world object references in the error fixing sub-task, $p = 0.11 > 0.017$.

The intention of our study was not to find which situation provides for spreadsheet authors to locate more errors. However, many participants had trouble identifying errors of **error type G** (“a wrong problem domain formula”). In particular, four participants (33%) literally wrote down “no error identified” when not using using the tool and only one participant (8%) wrote down the same when using the tool. Seven participants did not literally write down “no error identified” but had problems in either of the sub-tasks.

In summary, 25% of errors of **error type G** were correctly described by participants when not using the tool while 75% errors of the same error type were correctly described by participants when using the tool. On the other hand, 33% of errors of the same error type were correctly fixed by participants when not using the tool while 75% errors of the same error type were correctly fixed by participants when using the tool. Figure 6.13 and Figure 6.14 present a summary of participants’ performance for all the error types in the error description sub-task and error fixing sub-task respectively.

Discussion

In the *error describing* sub-task, participants used mostly spreadsheet terms when describing an error in the without tool case (13.0%

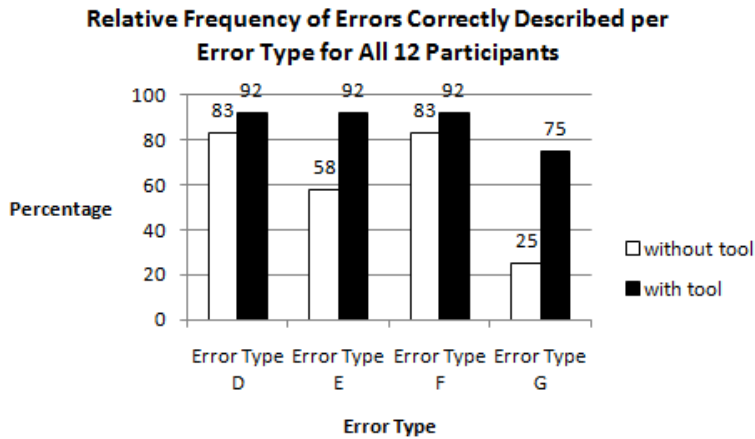


Figure 6.13: Bar chart illustrating relative frequencies of errors correctly described per error type without the tool and with the tool.

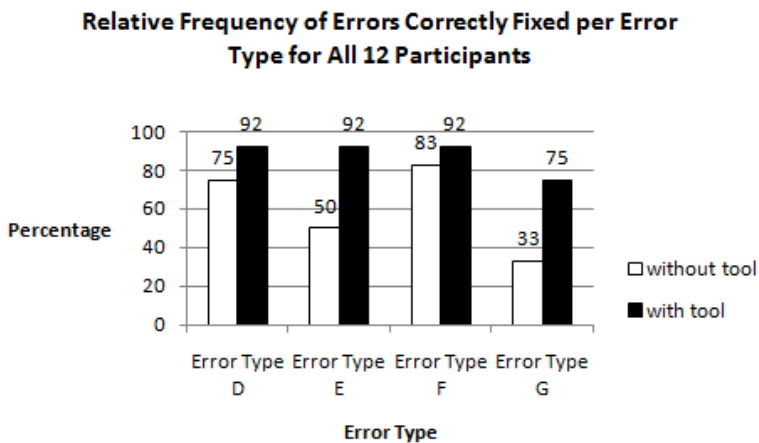


Figure 6.14: Bar chart illustrating relative frequencies of errors correctly fixed per error type without the tool and with the tool.

domain references vs 91.6% spreadsheet references). The use of domain terms is rather less evident. This is in line with the earlier study described in Chapter 5, i.e., with no tool support, it was found that spreadsheet authors mainly use the spreadsheet model in describing errors in spreadsheets.

However, in the study in Chapter 3, participants using plain Microsoft Excel, i.e. without tool support, verbally described errors using 52.2% domain references vs 51.3% spreadsheet references; in the study in Chapter 5, participants without tool support described errors (through written transcript) using 23.3% domain references vs 92.8% spreadsheet references; and in the study reported in this chapter (Chapter 6), participants without tool support described errors (through written transcript) using 13.0% domain references vs 91.6% spreadsheet references. The results of the without tool situation in the studies in Chapter 5 and and this chapter (Chapter 6) seem to agree although in the study in Chapter 5, the participants were not explicitly told to write down why particular cells were erroneous. However, both are written transcripts. This is however different if we compare with the verbal transcript of the study in Chapter 3. The spreadsheet references in the verbal transcript seem to be roughly the same as the domain references. This is in contrast to written transcripts where spreadsheet references are clearly more prominent than domain references.

This could be explained in several ways. First, the nature of the transcript could have an effect. We put forward that when a person talks aloud (verbal transcript) they become more verbose to make more sense to the listening person while when writing down their thoughts (written transcript), they write straight to the point just to capture the very essence of what they are thinking hence the reduction in domain references. As Horowitz and Newman (1964) corroborates - speaking and writing represent different strata of a person and although both functions funnel thought processes, speech includes more feeling and more "first thoughts" and impulsivity, whereas writing samples more of the intellectualized and rational and deliberate person. Second, in the study in Chapter 3, partic-

Participants were working with their own spreadsheets whereas in the other two studies, spreadsheets were originally created by others. We put forward that in the studies in Chapter 5 and this chapter (Chapter 6), the participants had to first comprehend the spreadsheet and – as they were searching for errors – they probably did this by looking at the formulae, hence starting their work with the spreadsheet model and resorting to the domain model only to compare the spreadsheet model with the domain model. In the study in Chapter 3, they had a very good domain model of the spreadsheet from the very beginning, and hence could utilize it more. Thus, the ownership of a spreadsheet (whether being the original author or a later debugger) probably affects the roles of the three mental models, especially when starting to work with a spreadsheet.

There could also be other uncontrolled factors that could affect the differences. Thus the mental models in the without tool support situation would not be inherently different in their content, but their prominence probably varies. Indeed, in all error description tasks without the tool, the prominence of the spreadsheet model is either the same as the domain model (verbal transcript) or is more than the domain model (written transcript) but it is not less prominent than the domain model.

On the other hand, in this study, with the paradigm shift tool, the situation changes significantly - the participants use mostly domain terms when describing an error in the tool case while the use of spreadsheet terms is rather less evident (compare 13.0% domain references vs 91.6% spreadsheet references in the without tool case with 65.9% domain references vs 42.3% spreadsheet references in the tool case). In the tool case, the domain reference usage not only increased to 65.9% (from 13.0% without the tool case) but also overshadowed the 42.3% spreadsheet reference usage. Thus the tool not only increases the usage of the domain model but also makes the domain model to eclipse the usage of the spreadsheet model. Thus the tool promotes a shift from having the spreadsheet model always dominant (as in the without tool case or traditional spreadsheet paradigm) to having the domain model dominant over the spread-

sheet model in the tool case. Thus the tool promotes a paradigm shift in the error description sub-task. And indeed looking at the transcripts, for example, without the tool, one participant described a “formula incorrectly referencing to neighbouring range” error as “instead of adding from C16 to I16, it has wrongly added from C15 to I15”. With the tool in use, the participant however described a similar (but not same error) in the other spreadsheet as “Profit after taxation for 1992 is wrongly added, that is, it is picking items from 1993”. The tool thus has an effect on the spreadsheet authors’ mental models when describing or locating errors in a spreadsheet.

The tool seems to provide a platform for a spreadsheet author to describe an error in terms of the problem domain rather than the traditional low-level cell references and hence providing a paradigm shift in how spreadsheet authors normally describe errors in spreadsheets. This effect could be positive because as we found in Chapter 5, spreadsheet authors located more errors with the domain terms spreadsheet visualization tool than without the tool. Although the intention of this study was not to find which situation provides for spreadsheet authors to locate more errors, with the tool, participants had relative ease in finding out why a particular cell was erroneous. Particularly, without the tool, many participants had trouble identifying why a “wrong problem domain formula” (**error type G**) erroneous cell was as such. This was however not the case with the tool. For example, in one error, “Values of Shares” were erroneously added to “Number of Shares” which obviously does not make domain sense. As reported above, 25% of errors of **error type G** were correctly described by participants when not using the tool while 75% errors of the same error type were correctly described by participants when using the tool (see Figure 6.13). By displaying formulae in domain terms, the tool it seems, helps the study participants to think mainly in terms of the problem domain and thus to discern easily that a formula does not make domain sense.

In the *error fixing* sub-task, participants used mostly spreadsheet terms when fixing an error in the without tool case (6.4% domain

references vs 97.5% spreadsheet references). The use of domain terms is rather less evident. This is in line with the study described in Chapter 3 that was conducted in a similar situation, i.e., with no tool support, it was found that spreadsheet authors mainly use the spreadsheet model in fixing errors in spreadsheets. However, in the study in Chapter 3, participants using plain Microsoft Excel (i.e. without the tool) when fixing errors used 36.5% domain references vs 64.5% spreadsheet references; and in the study in this chapter (Chapter 6), participants without tool support when fixing errors (through written transcript) used 6.4% domain references vs 97.5% spreadsheet references. The spreadsheet references in the study in Chapter 3 (verbal transcript) seem subdued while the domain references increase. This is in contrast to the study in this chapter (written transcript) where spreadsheet references are clearly more prominent than domain references. These differences could be explained in a similar way as in the error describing sub-task.

In either case, the spreadsheet model is always more prominent than the domain model. Thus the mental models in the without the tool case would not be inherently different in their content, but their prominence probably varies and indeed in all error fixing tasks without the tool, the spreadsheet model is always prominent over the domain model whether in a verbal transcript or a written transcript.

On the other hand, in this study, with the paradigm shift tool, the situation radically changes - the participants use mostly domain terms when fixing an error in the tool case while the use of spreadsheet terms is rather less evident (compare 6.4% domain references vs 97.5% spreadsheet references in the without tool case with 76.4% domain references vs 32.5% spreadsheet references in the tool case). In the tool case, the domain reference usage not only increased to 76.4% (from 6.4% without the tool case) but also overshadowed the 32.5% spreadsheet reference usage. Thus the tool not only increases the usage of the domain model but also makes the domain model to eclipse the usage of the spreadsheet model. Thus the tool promotes a shift from having the spreadsheet model always dominant

(in the without tool case or traditional spreadsheet paradigm) to having the domain model dominant over the spreadsheet model in the tool case. Thus the tool also promotes a paradigm shift in the error fixing sub-task.

The shift in the roles of the mental models however seems to vary depending on the task at hand. The shifting is larger in the error fixing sub-task than in the error description sub-task (see Figure 6.11 and Figure 6.12 in the with tool case). The difference in the prominence of the domain model and the spreadsheet model in the error fixing task is larger than in the error description sub-task – 32.5% domain references vs 76.4% spreadsheet references (difference of 43.9 percentage points) in error fixing while 42.3% domain references vs 65.9% spreadsheet references (difference of 23.6 percentage points) in error descriptions. We put forward that this could be the case because there is also a paradigm shift in the way one also does the actual fixing of errors (a physical manifestation of the paradigm shift in that instead of making a correction on a spreadsheet formula the person now edits a domain narrative and thus further enforcing a paradigm shift). Nevertheless, a person will still have to verify that fixing an error in domain terms also resulted in the right change in the spreadsheet model. Thus as long as one is in the current traditional spreadsheet environment, the spreadsheet model can not be completely eliminated in the spreadsheet author's mind. However, we envisage that in a spreadsheet paradigm where cell references are not used, the spreadsheet model, as defined in our context, could vanish totally.

In addition to changing or shifting in the roles of the mental models in the two sub-tasks, the tool also improves the mapping between the domain mental model and the spreadsheet mental model as the difference in the prominence of the domain mental model and the spreadsheet mental model in the tool use scenario is much smaller than in the without tool scenario (see Figure 6.11 and Figure 6.12). The mapping is however stronger in the error description task as compared to the error fixing scenario as the difference in the prominence of the domain model and the spreadsheet scenario

is smaller in the error description task than in the error fixing task. We however expect that with further usage of the tool, the usage of spreadsheet terms would decrease even further and use of domain terms would further increase. The improved mapping thus seems to be temporary as we expect the domain model to completely eclipse the spreadsheet model with further usage of the tool.

The tool, therefore, does not only promote a shift in the roles of mental models but also improves the mapping temporarily between the domain model and spreadsheet model although in varying degrees depending on the task. This shift in mental models is significant yet the study participants traditionally (without use of tool) describe and fix spreadsheet errors mainly in spreadsheet terms. Therefore, again, we could expect that with further usage of the paradigm shift tool, the usage of spreadsheet terms would decrease even further and the use of domain terms would further increase. A novice starting to use spreadsheets with the tool (i.e., a newcomer in spreadsheets) would probably, therefore, not talk in spreadsheet terms at all and therefore talk in domain terms only. These results further solidify our case for the need to have a paradigm shift in spreadsheets - a move from the current traditional spreadsheet paradigm to a paradigm that is closer to the problem domain.

Threats to Validity of Study

Just as in the study reported in Chapter 5, the spreadsheets that the participants used in this study were not created by the participants themselves and as such there could be an effect of unfamiliarity with the spreadsheets on the tasks at hand. However, the spreadsheets were quite simple and from the accounting field which the participants themselves belong to. Moreover, spreadsheet users frequently use spreadsheets created by others (Mittermeier & Clermont, 2002). As such, using spreadsheets created by others could help to realistically evaluate the effect of the tool on spreadsheet authors.

The tool itself is clumsy as it is just a prototype tool hence could affect the validity of the study. However, for example, in the study

we were not interested in the actual steps on how the participants could fix an error using the tool or without the tool but rather a participant's general understanding of an error fix. Thus we were not interested in the usability of the tool but rather to gauge the mental models of the participants when doing the sub-tasks.

The tool in its current form does not fully support well a distinction between relative and absolute references particularly if one wants to specify an absolute reference in the domain narrative. This could bring confusion to a spreadsheet author and hence affect the study. However, the spreadsheets we used in this study did not have absolute references hence this confusion could not arise.

6.3 CONCLUSION

We reported a paradigm shift that we are proposing in spreadsheet creation and debugging. We have thus demonstrated a prototype of a spreadsheet visualization tool that allows spreadsheet authors to fix errors in a spreadsheet in domain terms which is currently not easily done in the traditional spreadsheet. This demonstration answers the research question **RQ8** – *Is it possible to construct a tool that is based on a different paradigm rather than the current traditional spreadsheet paradigm?* Our empirical evaluation of the prototype paradigm shift tool showed that the tool promotes spreadsheet authors to think more in domain terms in a way that overshadows the way spreadsheet authors traditionally think when they are describing errors and even when fixing errors. This finding answers the research question **RQ9** – *What are the effects of such a tool on spreadsheet authors' mental models?*

These findings provide evidence that it is possible to have a paradigm shift in spreadsheets from the traditional spreadsheet paradigm to a paradigm where spreadsheet authors express themselves more at a higher problem domain level than just at a spreadsheet specific low-level. And this paradigm shift could be advantageous as the study in Chapter 5 showed that with formulae expressed in domain terms, participants were able to locate more er-

rors than in the traditional spreadsheet paradigm.

7 Conclusion

7.1 A REFLECTION ON THE RESEARCH OBJECTIVES

In this research work, we presented one way in which the spreadsheet error problem could be dealt with. In particular, we chose to attack this problem, by studying spreadsheet authors' mental models in order to better understand why the spreadsheet process is so error-prone and to be able to devise new tools that better correspond to spreadsheet authors' mental processes. We chose to study spreadsheet authors' mental models because it is a common assertion that humans have mental models of the systems they interact with and it is difficult to explain many aspects of human behaviour without resorting to a construct such as mental models (Rouse & Morris, 1986). And as such, we argue that it is important to first of all understand what types of mental models spreadsheet authors possess when they are doing different spreadsheet process activities. In this work, therefore, we applied the theory of mental models to come up with a proposed solution to the spreadsheet error and comprehension problem.

For each of the *three* specific objectives we had in this research work, we thus now give summarized answers to the specific research questions we had.

For the *first objective*, i.e., investigating and characterizing mental models of spreadsheet authors as they are doing various spreadsheet tasks, and particularly, when they are explaining and debugging their spreadsheets, the corresponding research questions and their summarized answers are as follows:

RQ1 – *Do spreadsheet authors have several mental models when they are doing various spreadsheet tasks, and particularly, when they are explaining and debugging their spreadsheets?*

Based on the empirical study we conducted, presented in Chapter 3, we characterized spreadsheet authors to have (at least) three

mental models of a spreadsheet: the *real world model* that comprises general knowledge of the world around us; the *domain model* that represents knowledge of the problem domain and the functionality of the spreadsheet in problem domain or application terms; and the *spreadsheet model* that codes the expressions and data relationships in the spreadsheet.

RQ2 – *What are the roles of mental models in spreadsheet authors when they are doing various spreadsheet tasks, and particularly, when they are explaining and debugging their spreadsheets?*

Based on the empirical study we conducted, presented in Chapter 3, we found that when explaining a spreadsheet, the real-world and domain mental models are prominent and the spreadsheet model is less evident, but when locating and fixing error, one must constantly switch back and forth between the domain model and the spreadsheet model, which requires frequent use of the mapping between problem domain concepts and their spreadsheet model counterparts.

RQ3 – *What are the implications of mental models in spreadsheet authors on spreadsheet authoring and debugging tools?*

Based on the empirical study we conducted, presented in Chapter 3, we inferred that a tool intended to aid in comprehension and debugging of spreadsheets should make prominent real-world and problem domain concepts and *map* them *easily* to spreadsheet-specific details.

For the *second objective*, i.e., development of a spreadsheet visualization tool that demonstrates that it is possible to devise spreadsheet authoring and debugging tools that are easy to use, efficient and correspond to spreadsheet authors' mental models of spreadsheets, the corresponding research questions we had for this objective and their summarized answers are as follows:

Conclusion

RQ4 – *Is it possible to develop a spreadsheet understanding and debugging tool that relieves users from spreadsheet details and lets them utilize more of their mental model of the application domain?*

This question was covered in Chapter 4 and Chapter 5. In Chapter 4, we gave a description of the domain terms spreadsheet visualization tool that we developed. The tool displays formulae in higher level symbolic names or domain/real-world terms and referenced cells are automatically highlighted. In Chapter 5, we reported that we found that the tool increased the role of the domain model while significantly reducing the role of the spreadsheet model when locating errors.

RQ5 – *Can the tool be learned easily by users?*

This question was covered in Chapter 5 where we reported that overall, study participants performed well on both the learnability tasks that we gave them. And as such, we concluded that the tool was easy to learn.

RQ6 – *Can the tool help users in debugging their spreadsheets?*

This question was covered in Chapter 5 where we reported that the tool helped study participants to locate more errors in spreadsheets although different aspects of the tool may be more helpful for some error types than others.

RQ7 – *Can spreadsheet authors be satisfied with the tool after using it?*

This question was also covered in Chapter 5 where we reported that overall, the subjective opinion of the participants on the tool shows that, generally, participants found the tool useful and thus were satisfied with the tool.

For the *third objective*, i.e., putting forward a case if there is need to shift from the traditional spreadsheet paradigm to another paradigm

that can better reflect spreadsheet authors' mental models, we presented the same in Chapter 6. The corresponding research questions we had for this objective and their summarized answers are as follows:

RQ8 – *Is it possible to construct a tool that is based on a different paradigm rather than the current traditional spreadsheet paradigm?*

In Chapter 6 we presented a prototype of a spreadsheet visualization tool demonstrating spreadsheet debugging in a new paradigm and also gave out limitations of the tool. We called this tool a paradigm shift spreadsheet visualization tool. In particular, with the tool, the spreadsheet author could also fix errors in higher level domain terms than just in traditional low-level spreadsheet specific terms such as cell references. Currently, the traditional spreadsheet paradigm does not easily support this.

RQ9 – *What are the effects of such a tool on spreadsheet authors' mental models?*

We tackled this research question in Chapter 6 where we evaluated the prototype paradigm shift tool to gauge the effects of such a tool on spreadsheet authors' mental models. The results of the empirical evaluation of the tool showed that the tool promotes spreadsheet authors to think more in domain terms in a way that overshadows the way spreadsheet authors traditionally think when they are describing errors and even when fixing errors. In other words, when describing and fixing errors, the tool seems to make spreadsheet authors invoke the domain mental models much more than the spreadsheet mental model. This is in sharp contrast to having a prominent spreadsheet mental model when they are describing and fixing errors in the traditional spreadsheet environment. Thus the tool promotes a paradigm shift in the way spreadsheet authors think when debugging their spreadsheets.

7.2 OUR CONTRIBUTION

Hereby is a summary of our specific contributions, from this research work, in the area of spreadsheets:

- (i) We have characterized spreadsheet authors to have (at least) three mental models of a spreadsheet: the *real world model* that comprises general knowledge of the world around us; the *domain model* that represents knowledge of the problem domain and the functionality of the spreadsheet in problem domain or application terms; and the *spreadsheet model* that codes the expressions and data relationships in the spreadsheet. When explaining a spreadsheet, the real-world and domain mental models are prominent and the spreadsheet model is less evident, but when locating and fixing an error, one must constantly switch back and forth between the domain model and the spreadsheet model, which requires frequent use of the mapping between problem domain concepts and their spreadsheet model counterparts. These results suggest that a tool intended to aid in comprehension and debugging of spreadsheets should make prominent real-world and problem domain concepts and map them easily to spreadsheet-specific details.
- (ii) We have developed a new spreadsheet visualization tool (a domain terms spreadsheet visualization tool) that demonstrates that it is possible to devise spreadsheet authoring and debugging tools that are easy to use and that correspond to spreadsheet authors' mental models of spreadsheets. The tool translates traditional spreadsheet formulae into symbolic name based problem domain narratives and highlights referenced cells. The tool was found to be easy to learn and helped the study participants to locate more errors in spreadsheets. Furthermore, the tool increased the use of the domain mental model in error descriptions and seemed to improve the mapping between the spreadsheet model and the domain model.

Many tools anecdotally assume that symbolic names and formula translation can be useful to users. We thus believe that the empirical evaluation of our tool in locating errors in spreadsheets is one other contribution that empirically demonstrates the usefulness of translating traditional spreadsheet formulae into higher level problem domain terms in the spreadsheet process such as in debugging.

- (iii) We have also proposed a shift from the traditional spreadsheet paradigm to another paradigm that better reflects spreadsheet authors' mental models. In particular, when creating a spreadsheet, instead of spreadsheet authors defining their spreadsheets in traditional low-level spreadsheet specific terms only such as cell references, they could also do so by using higher level domain terms. And in debugging, when fixing errors in a spreadsheet, the spreadsheet author could also fix the errors in higher level domain terms than just in traditional low-level spreadsheet specific terms such as cell references. We thus developed a prototype tool (a paradigm shift spreadsheet visualization tool) to demonstrate that it is possible for spreadsheet authors to also fix spreadsheet errors in domain terms. Our empirical evaluation of the tool also showed that the tool promotes spreadsheet authors to think more in domain terms in a way that overshadows the way spreadsheet authors traditionally think when they are describing errors and even when fixing errors, and hence promoting a paradigm shift from traditional spreadsheets.

7.3 FUTURE WORK

The domain terms spreadsheet visualization tool presented in Chapter 4 gives out spreadsheet formula narrations in domain terms, but does not efficiently allow spreadsheet authors to enter or edit formulae using the same notation. Adding such feature to the tool would further decrease the need for the spreadsheet model and bring formula editing closer to the domain model. Though our

Conclusion

paradigm shift tool presented in Chapter 6 clumsily allows this, we believe that more work needs to be done to improve this feature and thus also we also need to comprehensively conduct its feasibility and usefulness. Moreover, ideally, spreadsheet authors should be allowed to create spreadsheets by writing formulae in domain terms. This will be part of our future work.

Another line of our future work will be to investigate how our proposed paradigm shift in spreadsheets and indeed other results of this research work, could be applied to other end-user programming environments.

Bennett Freinderson Kankuzi: Deficiencies in Spreadsheets - A Mental
Model Perspective

Bibliography

- Abraham, R., & Erwig, M. (2004). Header and unit inference through spatial analyses. In *Proceedings of the 2004 IEEE Symposium on Visual Languages and Human-Centric Computing* (pp. 165–172). Washington, DC, USA.
- Abraham, R., & Erwig, M. (2005a). Goal-directed debugging of spreadsheets. In *Proceedings of the 2005 IEEE Symposium on Visual Languages and Human-Centric Computing* (pp. 37–44). Washington, DC, USA.
- Abraham, R., & Erwig, M. (2005b). How to communicate unit error messages in spreadsheets. In *Proceedings of the First Workshop on End-User Software Engineering* (pp. 1–5). New York, NY, USA.
- Allen, R. B. (1997). Mental models and user models. *Handbook of Human-Computer Interaction*, 1, 49–63.
- Apple Inc. (2013). *Numbers for Mac*. Retrieved from <http://www.apple.com/mac/numbers/>
- Ayalew, Y. (2007). A user-centred approach for testing spreadsheets. *International Journal of Computing and ICT Research*, 1(1), 77–85.
- Ayalew, Y., Clermont, M., & Mittermeier, R. T. (2000). Detecting errors in spreadsheets. In *Proceedings of the 1st European Spreadsheet Risks Interest Group Symposium* (pp. 51–62). London, UK.
- Ayalew, Y., & Mittermeier, R. T. (2003). Spreadsheet debugging. In *Proceedings of the 4th European Spreadsheet Risks Interest Group Symposium*. Dublin, Ireland.
- Ballinger, D., Biddle, R., & Noble, J. (2003). Spreadsheet structure inspection using low level access and visualisation. In *Proceedings of the Fourth Australasian Conference on User Interfaces* (pp. 91–94). Darlinghurst, Australia.
- Belo, O., Cunha, J., Fernandes, J. P., Mendes, J., Pereira, R., & Saraiva, J. (2013). QuerySheet: A bidirectional query environment for model-driven spreadsheets. In *Proceedings of the*

- 2013 *IEEE Symposium on Visual Languages and Human-Centric Computing* (pp. 199–200). Washington, DC, USA.
- Bergantz, D., & Hassell, J. (1991). Information relationships in PROLOG programs: How do programmers comprehend functionality? *International Journal of Man-Machine Studies*, 35(3), 313–328.
- Blackwell, A. (2002). What is programming? In J. Kuljis, L. Baldwin, & R. Scoble (Eds.), *Proceedings of the 14th Annual Workshop of the Psychology of Programming Interest Group* (pp. 204–218). London, UK.
- Borgman, C. L. (1985). The user's mental model of an information retrieval system. In *Proceedings of the 8th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval* (pp. 268–273). New York, NY, USA.
- Brooks, R. (1983). Towards a theory of the comprehension of computer programs. *International Journal of Man-Machine Studies*, 18(6), 543–554.
- Burnett, M. (2009). What is end-user software engineering and why does it matter? In *End-User Development* (pp. 15–28). Springer.
- Burnett, M., Atwood, J., Djang, R. W., Gottfried, H., Reichwein, J., & Yang, S. (2001). Forms/3: A first order visual language to explore the boundaries of the spreadsheet paradigm. *Journal of Functional Programming*, 11(2), 155–206.
- Burnett, M., Cook, C., Pendse, O., Rothermel, G., Summet, J., & Wallace, C. (2003). End-user software engineering with assertions in the spreadsheet paradigm. In *Proceedings of the 25th International Conference on Software Engineering* (pp. 93–103). Washington, DC, USA.
- Burnett, M., Cook, C., & Rothermel, G. (2004). End-user software engineering. *Communications of the ACM*, 47(9), 53–58.
- Byckling, P., Kuittinen, M., Nevalainen, S., & Sajaniemi, J. (2004). An inter-rater reliability analysis of Good's program summary analysis scheme. In *Proceedings of the 16th Annual Workshop of the Psychology of Programming Interest Group* (pp. 170–184). Carlow, Ireland.

Bibliography

- Carroll, J. M., Olson, J. R., & Anderson, N. S. (1987). *Mental models in human-computer interaction: Research issues about what the user of software knows* (No. 12). The National Academies.
- Chadwick, D., Knight, B., & Rajalingham, K. (2001). Quality control in spreadsheets: A visual approach using color codings to reduce errors in formulae. *Software Quality Journal*, 9(2), 133–143.
- Chambers, C., & Scaffidi, C. (2010). Struggling to excel: A field study of challenges faced by spreadsheet users. In *Proceedings of the 2010 IEEE Symposium on Visual Languages and Human-Centric Computing* (pp. 187–194). Washington, DC, USA.
- Chen, T. Y., Kuo, F.-C., & Zhou, Z. Q. (2005). An effective testing method for end-user programmers. In *Proceedings of the First Workshop on End-user Software Engineering* (pp. 1–5). New York, NY, USA.
- Chen, Y., & Chan, H. C. (2000a). An exploratory study of spreadsheet debugging processes. In *The Pacific Asia Conference on Information Systems* (pp. 143 – 155). HongKong, China.
- Chen, Y., & Chan, H. C. (2000b). Visual checking of spreadsheets. In *Proceedings of the 1st European Spreadsheet Risks Interest Group Symposium* (pp. 75–85). London, UK.
- Clermont, M. (2005). Heuristics for the automatic identification of irregularities in spreadsheets. In *Proceedings of the First Workshop on End-user Software Engineering* (pp. 1–6). New York, NY, USA.
- Clermont, M., Hanin, C., & Mittermeier, R. T. (2002). A spreadsheet tool evaluated in an industrial context. In *Proceedings of the 3rd European Spreadsheet Risks Interest Group Symposium* (pp. 35–46). Cardiff, Wales.
- Cohen, L., Manion, L., & Morrison, K. (2013). *Research methods in education*. Routledge.
- Corritore, C. L., & Wiedenbeck, S. (1991). What do novices learn during program comprehension? *International Journal of Human-Computer Interaction*, 3(2), 199–222.
- Cunha, J., Erwig, M., & Saraiva, J. (2010). Automatically infer-

- ring classsheet models from spreadsheets. In *Proceedings of the 2010 IEEE Symposium on Visual Languages and Human-Centric Computing* (pp. 93–100). Washington, DC, USA.
- Davis, J. S. (1996). Tools for spreadsheet auditing. *International Journal of Human-Computer Studies*, 45(4), 429–442.
- Desimone, R., & Duncan, J. (1995). Neural mechanisms of selective visual attention. *Annual Review of Neuroscience*, 18(1), 193–222.
- Doyle, J. K., & Ford, D. N. (1998). Mental models concepts for system dynamics research. *System Dynamics Review*, 14(1), 3–29.
- Duggirala, P. (2012). *Excel auditing functions [spreadsheet risk management]*. Retrieved from <http://chandoo.org/wp/2012/01/18/excel-auditing-functions/>
- Ebrahimi, A., & Schweikert, C. (2006). Empirical study of novice programming with plans and objects. *ACM SIGCSE Bulletin*, 38(4), 52–54.
- Engels, G., & Erwig, M. (2005). ClassSheets: Automatic generation of spreadsheet applications from object-oriented specifications. In *Proceedings of the 20th IEEE/ACM International Conference on Automated Software Engineering* (pp. 124–133). New York, NY, USA.
- Erwig, M., Robin, A., Irene, C., & Steve, K. (2005). Automatic generation and maintenance of correct spreadsheets. In *Proceedings of the 27th IEEE/ACM International Conference on Software Engineering* (p. 136-145). Washington, DC, USA.
- Fisher, M., Cao, M., Rothermel, G., Cook, C. R., & Burnett, M. (2002). Automated test case generation for spreadsheets. In *Proceedings of the 24th International Conference on Software Engineering* (pp. 141–153). New York, NY, USA.
- Fisher, M., & Rothermel, G. (2005). The EUSES spreadsheet corpus: A shared resource for supporting experimentation with spreadsheet dependability mechanisms. In *Proceedings of the First Workshop on End-User Software Engineering* (pp. 1–5). New York, NY, USA.
- Galletta, D. F., Hartzel, K. S., Johnson, S., Joseph, J., & Rustagi, S.

Bibliography

- (1996). An experimental study of spreadsheet presentation and error detection. In *Proceedings of the 29th Hawaii International Conference on System Sciences (HICSS): Decision Support and Knowledge-Based Systems* (pp. 336–345). Washington, DC, USA.
- Glass, R. L. (1994). The software-research crisis. *IEEE Software*, 11(6), 42–47.
- Good, J. (1999). *Programming paradigms, information types and graphical representations: Empirical investigations of novice program comprehension*. (Unpublished doctoral dissertation). The University of Edinburgh College of Science and Engineering: The School of Informatics.
- Good, J., & Brna, P. (2004). Program comprehension and authentic Measurement: A scheme for analysing descriptions of programs. *International Journal of Human-Computer Studies*, 61(2), 169–185.
- Goodwin, C. J. (2009). *Research in psychology: Methods and design*. John Wiley & Sons.
- Google Inc. (2013). *Named and protected ranges*. Retrieved from <https://support.google.com/drive/answer/63175?hl=en>
- Götschi, T., Sanders, I., & Galpin, V. (2003). Mental models of recursion. *ACM SIGCSE Bulletin*, 35(1), 346–350.
- Green, T. R. G. (1990). The nature of programming. In J. M. Hoc, T. R. G. Green, R. Samurcay, & D. J. Gillmore (Eds.), *Psychology of Programming* (pp. 21–44). Academic Press.
- Green, T. R. G., & Navarro, R. (1995). Programming plans, imagery, and visual programming. In *INTERACT* (pp. 139–144). London, UK.
- 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 Human-Computer Studies*, 40(6), 1033–1066.
- Hermans, F., Pinzger, M., & van Deursen, A. (2011). Breviz: Visualizing spreadsheets using dataflow diagrams. In *Proceedings*

of the 12th European Spreadsheet Risks Interest Group Symposium.
London, UK.

- Hermans, F., Pinzger, M., & van Deursen, A. (2012). Detecting and visualizing inter-worksheet smells in spreadsheets. In *Proceedings of the 2012 International Conference on Software Engineering* (pp. 441–451).
- Hoadley, C. M., Linn, M. C., Mann, L. M., & Clancy, M. J. (1996). When, why and how do novice programmers reuse code. In *Empirical Studies of Programmers: Sixth Workshop* (pp. 109–129). Norwood, NJ, USA.
- Hofer, B., Riboira, A., Wotawa, F., Abreu, R., & Getzner, E. (2013). On the empirical evaluation of fault localization techniques for spreadsheets. In *Fundamental Approaches to Software Engineering* (pp. 68–82). Springer.
- Horowitz, M. W., & Newman, J. B. (1964). Spoken and written expression: An experimental analysis. *The Journal of Abnormal and Social Psychology*, 68(6), 640.
- Hughes, C., & Buckley, J. (2004). Evaluating algorithm animation for concurrent systems: A comprehension-based approach. In *Proceedings of the 16th Annual Workshop of the Psychology of Programming Interest Group* (pp. 193–205). Carlow, Ireland.
- Igarashi, T., Mackinlay, J. D., Chang, B. W., & Zellweger, P. T. (1998). Fluid visualization of spreadsheet structures. In *Proceedings of the IEEE Symposium on Visual Languages* (pp. 118–125). Washington, DC, USA.
- Iivari, J. (2007). A paradigmatic analysis of information systems as a design science. *Scandinavian Journal of Information Systems*, 19(2).
- JabSoft Ltd. (2013). *Spreadsheet Auditor for Excel*. Retrieved from <http://www.jabsoft.com/>
- Jannach, D., & Schmitz, T. (2014). Model-based diagnosis of spreadsheet programs: A constraint-based debugging approach. *Automated Software Engineering*, 1–40.
- Jannach, D., Schmitz, T., Hofer, B., & Wotawa, F. (2014). Avoiding, finding and fixing spreadsheet errors – A survey of auto-

Bibliography

- mated approaches for spreadsheet QA. *Journal of Systems and Software*.
- Jimoyiannis, A. (2013). Using SOLO taxonomy to explore students' mental models of the programming variable and the assignment statement. *Themes in Science and Technology Education*, 4(2), pp-53.
- Johnson-Laird, P. N. (1983). *Mental models*. Cambridge University Press.
- Johnson-Laird, P. N. (2010). Mental models and human reasoning. *Proceedings of the National Academy of Sciences*, 107(43), 18243–18250.
- Kankuzi, B., & Ayalew, Y. (2008a). An MCL algorithm based technique for comprehending spreadsheets. In *20th Annual Conference of the Psychology of Programming Interest Group* (pp. 4–14). Lancaster, UK.
- Kankuzi, B., & Ayalew, Y. (2008b). A user-centered graph-based visualization for spreadsheets. In *Proceedings of the 4th International Workshop on End-User Software Engineering* (pp. 86–90). New York, NY, USA.
- Kasanen, E., & Lukka, K. (1993). The constructive approach in management accounting research. *Journal of Management Accounting Research*(5), 243–264.
- Ko, A. J. (2007). Barriers to successful end-user programming. In M. H. Burnett, G. Engels, B. A. Myers, & G. Rothermel (Eds.), *End-User Software Engineering*. Internationales Begegnungs- und Forschungszentrum fuer Informatik (IBFI).
- Ko, A. J., Abraham, R., Beckwith, L., Blackwell, A., Burnett, M., Erwig, M., ... Wiedenbeck, S. (2011). The state of the art in end-user software engineering. *ACM Computing Surveys*, 43(3), 21.
- Kulesz, D. (2011). From good practices to effective policies for preventing errors in spreadsheets. In *Proceedings of European Spreadsheet Risk Interest Group 2011 Conference*. London, UK.
- Kurland, D. M., & Pea, R. D. (1985). Children's mental models of recursive LOGO programs. *Journal of Educational Computing*

Research, 1(2), 235–243.

- Labro, E., & Tuomela, T.-S. (2003). On bringing more action into management accounting research: Process considerations based on two constructive case studies. *European Accounting Review*, 12(3), 409–442.
- Lazar, J., Feng, J. H., & Hochheiser, H. (2010). *Research methods in human-computer interaction*. Wiley.
- Letovsky, S. (1987). Cognitive processes in program comprehension. *Journal of Systems and Software*, 7(4), 325–339.
- Lukka, K. (2000). The key issues of applying the constructive approach to field research. In T. Reponen (Ed.), *Management Expertise for the New Millenium. In Commemoration of the 50th Anniversary of the Turku School of Economics and Business Administration* (pp. 113–128). Turku, Finland.
- Ma, L., Ferguson, J., Roper, M., & Wood, M. (2007). Investigating the viability of mental models held by novice programmers. *ACM SIGCSE Bulletin*, 39(1), 499–503.
- Markman, A. B., & Gentner, D. (2001). Thinking. *Annual Review of Psychology*, 52(1), 223–247.
- McKeever, R., McDaid, K., & Bishop, B. (2009). An exploratory analysis of the impact of named ranges on the debugging performance of novice users. In *Proceedings of European Spreadsheet Risks Interest Group 2009 Conference*. Paris, France.
- McKenzie, S. (2007). *Empirical research methods for human-computer interaction – 2007 CHI course notes*. Retrieved from <http://www.yorku.ca/mack/CourseNotes.pdf>
- Miller, D., Miller, G., & Parrondo, L. M. (2010). Sumwise: A smarter spreadsheet. In *Proceedings of the European Spreadsheet Risks Interest Group 2010 Conference*. London, UK.
- Mittermeier, R., & Clermont, M. (2002). Finding high-level structures in spreadsheet programs. In *Proceedings of the Ninth Working Conference on Reverse Engineering* (pp. 221–232). Washington, DC, USA.
- Myers, B. A., Burnett, M., & Rosson, M. B. (2005). End users creating effective software. In *CHI'05 Extended Abstracts on Human*

Bibliography

- Factors in Computing Systems* (pp. 2047–2048). New York, NY, USA.
- Myers, B. A., Ko, A. J., & Burnett, M. M. (2006). Invited research overview: End-user programming. In *CHI '06 Extended Abstracts on Human Factors in Computing Systems* (pp. 75–80). New York, NY, USA.
- Nardi, B. A. (1993). *A small matter of programming: Perspectives on end-user computing*. The MIT Press.
- Nardi, B. A., & Zарmer, C. L. (1993). Beyond models and metaphors: Visual formalisms in user interface design. *Journal of Visual Languages and Computing*, 4, 5–33.
- Nardi, D., & Serrecchia, G. (1994). Automatic generation of explanations for spreadsheet applications. In *Proceedings of the Tenth Conference on Artificial Intelligence for Applications* (pp. 268–274). Washington, DC, USA.
- Navarro-Prieto, R., & Cañas, J. J. (2001). Are visual programming languages better? The role of imagery in program comprehension. *International Journal of Human-Computer Studies*, 54(6), 799–829.
- Nielsen, J. (1994). *Usability engineering*. Boston: AP Professional.
- Norman, D. (1983). Some observations on mental models. In D. Gentner & A. L. Stevens (Eds.), *Mental Models* (pp. 7–14). Lawrence Erlbaum Associates.
- O'Shea, P., & Exton, C. (2004). The application of content analysis to programmer mailing lists as a requirements method for a software visualisation tool. In *Proceedings of the 12th International Workshop on Software Technology and Engineering Practice* (pp. 10–14). Washington, DC, USA.
- Panko, R. R. (1998). What we know about spreadsheet errors. *Journal of Organizational and End User Computing*, 10(2), 15–21.
- Panko, R. R. (2000). Spreadsheet errors: What we know. What we think we can do. In *Proceedings of the European Spreadsheet Risk Interest Group Symposium*. Greenwich, UK.
- Panko, R. R., & Ordway, N. (2005). Sarbanes-Oxley: What about all the spreadsheets? Controlling for errors and fraud in finan-

- cial reporting. In *Proceedings of the European Spreadsheet Risks Interest Group 2005 Conference*. London, UK.
- Pennington, N. (1987). Stimulus structures and mental representations in expert comprehension of computer programs. *Cognitive Psychology*, 19(3), 295–341.
- Piirainen, K. A., & Gonzalez, R. A. (2013). Seeking constructive synergy: Design science and the constructive research approach. In *Design Science at the Intersection of Physical and Virtual Design* (pp. 59–72). Springer.
- Powell, S. G., Baker, K. R., & Lawson, B. (2009). Impact of errors in operational spreadsheets. *Decision Support Systems*, 47(2), 126–132.
- Raffensperger, J. F. (2008). *The art of the spreadsheet*. Retrieved from <http://john.raffensperger.org/john/ArtOfTheSpreadsheet/>
- Ramalingam, V., LaBelle, D., & Wiedenbeck, S. (2004). Self-efficacy and mental models in learning to program. *ACM SIGCSE Bulletin*, 36(3), 171–175.
- Randolph, N., Morris, J., & Lee, G. (2002). A generalised spreadsheet verification methodology. In *Proceedings of the Twenty-Fifth Australasian Conference on Computer Science* (pp. 215–222). Darlinghurst, Australia.
- Rothermel, G., Burnett, M., Li, L., Dupuis, C., & Sheretov, A. (2001). A methodology for testing spreadsheets. *ACM Transactions on Software Engineering and Methodology*, 10(1), 110–147.
- Rouse, W. B., & Morris, N. M. (1986). On looking into the black box: Prospects and limits in the search for mental models. *Psychological Bulletin*, 100(3), 349.
- Ruthruff, J. R., Prabhakararao, S., Reichwein, J., Cook, C., Creswick, E., & Burnett, M. (2005). Interactive, visual fault localization support for end-user programmers. *Journal of Visual Languages and Computing*, 16(1-2), 3–40.
- Saariluoma, P., & Sajaniemi, J. (1989). Visual information chunking in spreadsheet calculation. *International Journal of Man-Machine Studies*, 30(5), 475–488.

Bibliography

- Sajaniemi, J. (2000). Modeling spreadsheet audit: A rigorous approach to automatic visualization. *Journal of Visual Languages & Computing*, 11(1), 49–82.
- Sajaniemi, J., & Kuittinen, M. (2005). An experiment on using roles of variables in teaching introductory programming. *Computer Science Education*, 15(1), 59–82.
- Sajaniemi, J., & Pekkanen, J. (1988). An empirical analysis of spreadsheet calculation. *Software: Practice and Experience*, 18(6), 583–596.
- Scaffidi, C., Shaw, M., & Myers, B. (2005). An approach for categorizing end user programmers to guide software engineering research. In *Proceedings of the First Workshop on End-user Software Engineering* (pp. 1–5). New York, NY, USA.
- Segal, J. (2005). Two principles of end-user software engineering research. In *Proceedings of the First Workshop on End-user Software Engineering* (pp. 1–5). New York, NY, USA.
- Seltman, H. J. (2012). *Experimental design and analysis*. Carnegie Mellon University.
- Seta, K., Ikeda, M., Kakusho, O., & Mizoguchi, R. (1997). Capturing a conceptual model for end-user programming: Task ontology as a static user model. In A. Jameson, C. Paris, & C. Tasso (Eds.), *User Modeling: Proceedings of the Sixth International Conference, UIM97* (p. 203-214). Vienna: Springer.
- Soloway, E., & Ehrlich, K. (1984). Empirical studies of programming knowledge. *IEEE Transactions on Software Engineering*(5), 595–609.
- Southern Cross Software. (2013). *Spreadsheet Detective*. Retrieved from <http://www.spreadsheetdetective.com/>
- Spreadsheet Innovations Ltd. (2013). *Spreadsheet Professional*. Retrieved from <http://www.spreadsheetinnovations.com/>
- Staggers, N., & Norcio, A. F. (1993). Mental models: Concepts for human-computer interaction research. *International Journal of Man-Machine Studies*, 38(4), 587–605.
- Stemler, S. (2001). An overview of content analysis. *Practical Assessment, Research & Evaluation*, 7(17), 137–146.

- Storey, M. A. D. (2005). Theories, methods and tools in program comprehension: Past, present and future. In *Proceedings of the 13th International Workshop on Program Comprehension* (pp. 181–191). Washington, DC, USA.
- Storey, M. A. D., Fracchia, F. D., & Müller, H. A. (1999). Cognitive design elements to support the construction of a mental model during software exploration. *Journal of Systems and Software, 44*(3), 171–185.
- The R Foundation. (2013). *The R project for statistical computing*. Retrieved from <http://www.r-project.org/>
- Tukiainen, M. (2001). Comparing two spreadsheet calculation paradigms: An empirical study with novice users. *Interacting with Computers, 13*(4), 427–446.
- von Mayrhauser, A., & Vans, A. M. (1995). Program comprehension during software maintenance and evolution. *Computer, 28*(8), 44–55.
- Wertheimer, M. (1938). *A source book of Gestalt psychology*. Routledge & Kegan Paul.
- Wiedenbeck, S., & Ramalingam, V. (1999). Novice comprehension of small programs written in the procedural and object-oriented styles. *International Journal of Human-Computer Studies, 51*(1), 71–87.
- Wilson, A., Burnett, M., Beckwith, L., Granatir, O., Casburn, L., Cook, C., ... Rothermel, G. (2003). Harnessing curiosity to increase correctness in end-user programming. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems* (pp. 305–312). New York, NY, USA.

Appendix A: Part of the spreadsheet used by participant no. 2 in the spreadsheet explanation task

Note: Confidential data has deliberately been blurred.

PROFIT/LOSS STATEMENT	<---1ST QUARTER SEPTEMBER--->			<---1ST QUARTER SEPTEMBER--->	
	ACTUAL	BUDGET	ACTUAL	Variance	Variance
	1st Quarter	1st Quarter	1st Quarter	Actual	Actual qtr Sept
	ended Sept	ended Sept	ended Sept	VS Budget	VS Actual qtr Sept
Revenue					
Passenger & Excess baggage (Net of commission paid)	3	3	3		
Cargo	1	1	1		
Fuel & Insurance surcharge	5	5	5		
Adhoc charters					
Total Revenue	9	9	9		
Direct Operating Costs					
Fuel and Oil	3	3	3		
Navigation	1	1	1		
Landing	1	1	1		
Handling	3	3	3		
Catering	1	1	1		
Expenses					
Maintenance (routine & major checks)	2	2	2		
Communication charges - Sita, galileo, amadeus, sabre	1	1	1		
Hire - adhoc	1	1	1		
lease & maintenance reserve	1	1	1		
Engine hire	6	6	6		
Total Direct Operating Costs	15	15	15		
GROSS CONTRIBUTION	(6)	(6)	(6)		
Fixed costs:					
Insurance	3	3	3		
Interest Charges	1	1	1		
Total Fixed Costs	4	4	4		
Net Contribution	(10)	(10)	(10)		
Other income					
Drop-offs	2	2	2		

Bennett Freinderson Kankuzi: Deficiencies in Spreadsheets - A Mental
Model Perspective

Appendix B: Transcript of an interview with participant no. 2 in the spreadsheet explanation task

Interviewer (Q): Bennett Kankuzi. Respondent (A): <name deleted>

Q:

Thanks a lot for accepting to participate in this study. First of all I would like to ask you on your background in creating spreadsheets. How many spreadsheets have you created so far?

A:

So many, because mostly the spreadsheets we create them for management accounts monthly, quarterly and sometimes half a year. so mostly we play around same sort of formatting.

Q:

Do you create the spreadsheets for yourself?

A:

Yes and and we try to modify it from other sources we try to create something to link to it, like where we get the inputs when I create it I just leave it to get the data direct from it.

Q:

So you created this spreadsheet for what?

A:

For quarterly management accounts

Q:

In general, do you encounter problems when you are creating spreadsheets?

A:

The challenges which we have in our spreadsheet, is that it is so big. It is like a moving one, because we have to add in additional months so sometimes when you are not careful that you are on which month you tend to get data from the other month.

***** PROGRAM SUMMARY ANALYSIS STARTS HERE *****
(focus is on answers from the participant)

Q:

So we have a spreadsheet in front of us here which you have circled groups of related cells. What is this spreadsheet all about?

A:

This spreadsheet is all about profit and loss for one quarter, for three months.

Q:

And then I can see you have circled groups of cells there. Would you mind explaining why you have circled them as such?

A:

I have circled them mostly on the top parts. This is like our main revenue with three lines I circled these cells vertically. They are related in that they are both revenue. On the first one, is the passenger revenue. Second one it's cargo, we carry cargo on our planes. The third column is like on the same passenger we charge fare, but in our sales internally we do apportion like fuel surcharges and sales surcharges for our passengers. So we try to split it to show as a percentage of our revenue and in a way this column it is related horizontally also because these are turned to passenger revenue. These are the same cargo revenue and fuel surcharge but they are also related like this one. These are actual for the quarter. This is budget for the quarter. Due to past experience, we do have also budget to compare how well we do and this column is related to this one in a way that it compares with the same quarter last year. We do also compare how we performed last year in the same quarter, so on these ones as well, we are saying they are related because they are like direct operating cost for us to operate. We have to have fuel, landing, handling, so they are

also related to this bottom line if they are overheads like salaries, personnel but we differentiate them.

Q:

That's why you have drawn this line that crosses over?

A:

Yeah, from direct operating cost to overheads, general overheads, the same as with these ones and these, they are related in a way that these are direct operating costs, these are actual and these are for the same quarter previously. And this is what we call other revenues I mean fixed cost, the insurance cost and the fixed cost we differentiate them from direct operating costs because these ones regardless of whether we operate or not, but we have to pay them. So in a way, they are related in their own, so these subtotal for the cost, that is the total direct operating cost, total fixed cost and total general overheads. They are also related because they are all our cost which have been subtracted from our revenues. But our revenues as well, they are categorised into two. The other ones are these ones where we call other income. Other income comprises of we call ticket block-outs. Ticket block-outs are like when passengers buy tickets from us we don't take it as a sale. To us is a liability because we haven't provided a service yet in that account. What will happen is, either people will be refunded or people will fly on another airline, those airlines will bill us. If they fly on our airline that's where these ones we recognise as our revenue. But the ticket drop-outs in the airlines we have sales. There are some tickets which are not refundable because they are discounted so with the passenger like yourself after six months but after normally a year, we say it is a drop-out because it's our revenue though we haven't provided a service but by agreement and other airlines also if they have not billed us we need specific time as specified by our time, we take it as our revenue so that's a drop out. So for <system name deleted>, it was a system of selling airline tickets where by <airline name deleted> was providing that service on behalf of <service provider name deleted> in <country name deleted>. This is where we were getting commission. Then there is this

management fees, which have two subsidiaries, the <company name deleted> and <company name deleted>. These are government grants, it's like being a government institution once in a while to give us grants so we group them on their own like they are related, like they are not part of our core business as opposed to ticket sales so we try to show them differently because they are not part of our business but it's still they are our revenues.

Q:

So I have also seen that they are other cells which you have not marked or grouped. For example, down there, on 'less depreciation' on air craft and non aircraft. Why did you not like group them or let them to be part of some groups? Was it deliberate?

A:

No, it was not deliberate it's only because I was doing it in a hurry, because they have also some relationships on their own.

Q:

And I have seen that you have have some groups of related cells which are overlapping in the sense that they belong to one group of related cells but you have also put them in another group of related cells, why is it so?

A:

It's so because they are like, what causes the overlapping, is like operating cost and other overheads or fixed cost they are part of the costs but because we tried to create different cells because of their nature.

Q:

Because of their nature? What do you mean by their nature?

A:

It means that we wanted to show that those costs without them we can't operate.

Q:

And for variance actual for September 2011, I have seen that you have also not grouped these? Was this also deliberate?

A:

No, it was not deliberate but I was somehow in a hurry. These as

Transcript of an interview with participant no. 2 in the spreadsheet
explanation task

well could be related some sort of in one way or the other. Like these variances, I do not know if I can be circling them now?

**** PROGRAM SUMMARY ANALYSIS ENDS HERE ****

Q:

You can leave them as such, it's no problem. I don't know if you have got any comments on spreadsheets before we close the interview?

A:

I do not have any but because I do a lot of work on spreadsheets, I intend to do a refresher course on spreadsheets because it seems to be a very important tool.

Q:

Thanks so much for your time.

Bennett Freinderson Kankuzi: Deficiencies in Spreadsheets - A Mental
Model Perspective

Appendix C: Sample object reference classification used in program summary analysis

Object reference classification used in program summary analysis of transcript for participant no. 2 in the spreadsheet explanation task

No.	Object reference	Real-world specific	Domain specific	Spreadsheet specific
1.	<i>profit and loss</i>	no	yes	no
2.	<i>one quarter</i>	no	yes	no
3.	<i>three months</i>	yes	yes	no
4.	<i>main revenue</i>	no	yes	no
5.	<i>three lines</i>	no	yes	no
6.	<i>cells</i>	no	no	yes
7.	<i>revenue</i>	yes	yes	no
8.	<i>passenger revenue</i>	no	yes	no
9.	<i>cargo</i>	yes	yes	no
10.	<i>cargo</i>	yes	yes	no
11.	<i>planes</i>	yes	no	no
12.	<i>third column</i>	no	no	yes

Continuation from previous page - Object reference classification used in program summary analysis of transcript for participant no. 2 in the spreadsheet explanation task

No.	Object reference	Real-world specific	Domain specific	Spreadsheet specific
13.	<i>passenger</i>	yes	yes	no
14.	<i>fare</i>	no	yes	no
15.	<i>sales</i>	yes	yes	no
16.	<i>fuel surcharges</i>	no	yes	no
17.	<i>sales surcharges</i>	no	yes	no
18.	<i>passengers</i>	yes	yes	no
19.	<i>percentage of our revenue</i>	no	yes	no
20.	<i>column</i>	no	no	yes
21.	<i>passenger revenue</i>	no	yes	no
22.	<i>cargo revenue</i>	no	yes	no
23.	<i>fuel surcharge</i>	no	yes	no
24.	<i>actual for the quarter</i>	no	yes	no
25.	<i>budget for the quarter</i>	no	yes	no
26.	<i>budget</i>	yes	yes	no
27.	<i>column</i>	no	no	yes
28.	<i>same quarter</i>	no	yes	no
29.	<i>last year</i>	yes	yes	no
30.	<i>direct operating cost</i>	no	yes	no
31.	<i>fuel</i>	yes	yes	no
32.	<i>landing</i>	no	yes	no

Continuation from previous page - Object reference classification used in program summary analysis of transcript for participant no. 2 in the spreadsheet explanation task

No.	Object reference	Real-world specific	Domain specific	Spreadsheet specific
33.	<i>handling</i>	no	yes	no
34.	<i>overheads</i>	no	yes	no
35.	<i>salaries</i>	yes	yes	no
36.	<i>personnel</i>	no	yes	no
37.	<i>direct operating costs</i>	no	yes	no
38.	<i>overheads</i>	no	yes	no
39.	<i>general overheads</i>	no	yes	no
40.	<i>direct operating costs</i>	no	yes	no
41.	<i>actual</i>	no	yes	no
42.	<i>same quarter</i>	no	yes	no
43.	<i>other revenues</i>	no	yes	no
44.	<i>fixed cost</i>	no	yes	no
45.	<i>insurance cost</i>	no	yes	no
46.	<i>fixed cost</i>	no	yes	no
47.	<i>direct operating costs</i>	no	yes	no
48.	<i>subtotal for the cost</i>	no	yes	no
49.	<i>total direct operating cost</i>	no	yes	no
50.	<i>total fixed cost</i>	no	yes	no
51.	<i>total general overheads</i>	no	yes	no
52.	<i>cost</i>	yes	yes	no

Continuation from previous page - Object reference classification used in program summary analysis of transcript for participant no. 2 in the spreadsheet explanation task

No.	Object reference	Real-world specific	Domain specific	Spread-sheet specific
53.	<i>revenues</i>	yes	yes	no
54.	<i>revenues</i>	yes	yes	no
55.	<i>other income</i>	no	yes	no
56.	<i>other income</i>	no	yes	no
57.	<i>ticket block-outs</i>	no	yes	no
58.	<i>ticket block-outs</i>	no	yes	no
59.	<i>passengers</i>	yes	yes	no
60.	<i>tickets</i>	yes	yes	no
61.	<i>a sale</i>	no	yes	no
62.	<i>liability</i>	no	yes	no
63.	<i>service</i>	yes	yes	no
64.	<i>account</i>	no	yes	no
65.	<i>people</i>	yes	no	no
66.	<i>people</i>	yes	no	no
67.	<i>airline</i>	yes	yes	no
68.	<i>airline</i>	yes	yes	no
69.	<i>airline</i>	yes	yes	no
70.	<i>revenue</i>	yes	yes	no
71.	<i>ticket drop-outs</i>	no	yes	no
72.	<i>airlines</i>	yes	yes	no

Continuation from previous page - Object reference classification used in program summary analysis of transcript for participant no. 2 in the spreadsheet explanation task

No.	Object reference	Real-world specific	Domain specific	Spreadsheet specific
73.	<i>sales</i>	yes	yes	no
74.	<i>passenger</i>	yes	yes	no
75.	<i>six months</i>	yes	yes	no
76.	<i>a year</i>	yes	yes	no
77.	<i>drop-out</i>	no	yes	no
78.	<i>revenue</i>	yes	yes	no
79.	<i>service</i>	yes	yes	no
80.	<i>agreement</i>	yes	yes	no
81.	<i>airlines</i>	yes	yes	no
82.	<i>specific time</i>	no	yes	no
83.	<i>time</i>	yes	yes	no
84.	<i>revenue</i>	yes	yes	no
85.	<i>drop-out</i>	no	yes	no
86.	<system name deleted>	no	yes	no
87.	<i>system</i>	yes	yes	no
88.	<i>airline tickets</i>	yes	yes	no
89.	<airline name deleted>	yes	yes	no
90.	<i>service</i>	yes	yes	no
91.	<service provider name deleted>	no	yes	no
92.	<country name deleted>	yes	yes	no

Continuation from previous page - Object reference classification used in program summary analysis of transcript for participant no. 2 in the spreadsheet explanation task

No.	Object reference	Real-world specific	Domain specific	Spread-sheet specific
93.	<i>commission</i>	no	yes	no
94.	<i>management fees</i>	no	yes	no
95.	<i>subsidiaries</i>	no	yes	no
96.	< <i>company name deleted</i> >	no	yes	no
97.	< <i>another company name deleted</i> >	no	yes	no
98.	<i>government grants</i>	no	yes	no
99.	<i>government institution</i>	yes	yes	no
100.	<i>grants</i>	no	yes	no
101.	<i>core business</i>	yes	yes	no
102.	<i>ticket sales</i>	yes	yes	no
103.	<i>business</i>	yes	yes	no
104.	<i>revenues</i>	yes	yes	no
105.	<i>relationships</i>	no	yes	no
106.	<i>operating cost</i>	no	yes	no
107.	<i>other overheads</i>	no	yes	no
108.	<i>fixed cost</i>	no	yes	no
109.	<i>costs</i>	no	yes	no
110.	<i>cells</i>	no	no	yes
111.	<i>nature</i>	no	yes	no
112.	<i>costs</i>	no	yes	no

Sample object reference classification used in program summary analysis

Continuation from previous page - Object reference classification used in program summary analysis of transcript for participant no. 2 in the spreadsheet explanation task

No.	Object reference	Real-world specific	Domain specific	Spreadsheet specific
113.	<i>variances</i>	no	yes	no

BENNETT FREINDERSON

KANKUZI

*Deficiencies in
Spreadsheets: A Mental
Model Perspective*

This thesis tackled the problem of errors in spreadsheets by studying mental models of spreadsheet authors performing various spreadsheet activities. Using a constructive research approach, we conducted three empirical studies and developed two spreadsheet visualization tools. Our results show the need for spreadsheet tools that reflect spreadsheet authors' mental models, and for a paradigm shift which allows spreadsheets to be developed in terms of a mental model of the application domain.



UNIVERSITY OF
EASTERN FINLAND

PUBLICATIONS OF THE UNIVERSITY OF EASTERN FINLAND

Dissertations in Forestry and Natural Sciences

ISBN: 978-952-61-1827-7 (PRINTED)

ISSNL: 1798-5668

ISSN: 1798-5668

ISBN: 978-952-61-1828-4 (PDF)

ISSNL: 1798-5668

ISSN: 1798-5676