A COMPARISON OF AUTOMATED AND MANUAL SPREADSHEET ERROR DETECTION

A THESIS PRESENTED IN PARTIAL FULFILLMENT
OF THE REQUIREMENTS FOR THE DEGREE OF
MASTER OF INFORMATION SYSTEMS

at
MASSEY UNIVERSITY, ALBANY, NEW ZEALAND

WARREN M ANDERSON
2004

Abstract

Research has confirmed that spreadsheet errors are very common, with spreadsheet error rates being comparable to those found in traditional software development. Whilst other methods exist, such as two or more people checking the spreadsheet manually, no single technique has been found that is able to detect the majority of spreadsheet errors. However, in 2001 a promising research project was conducted by Nixon and O'Hara which examined five spreadsheet audit tools and concluded that they were a useful aid for helping spreadsheet developers and auditors to locate certain kinds of spreadsheet errors.

To evaluate how well audit tools performed, both in terms of personnel and software, a sample spreadsheet, containing fifteen errors, was used to test the ability of sixty-five participants and eleven spreadsheet audit tools to detect the same errors. The results were then compared to analyse whether the two groups found the same or different errors

The results, whilst confirming the research hypothesis that audit tools and participants would locate different types of errors, further showed that the audit tools did not perform as well as the participants both in terms of the number of errors detected and the reasons for them. While these results supported previously held views that audit tools were unlikely to find the greatest number of the errors, they also confirmed the view of the 2001 study that spreadsheet audit tools were useful in helping to locate specific kinds of spreadsheet errors not detected by human participants. That finding leads to the conclusion that both spreadsheet tools and technically competent users are needed to ensure potentially costly errors are detected and corrected.

Acknowledgements

I would like to thank the following people for the help, support and encouragement given to me during my thesis year.

To my supervisor, Peter Blakey, Senior Lecturer in the Department of Information Systems at Massey University, for the amazing work he did reviewing my thesis material and for his unwavering encouragement and patience.

To Murray Black, Head of Mathematics at the Auckland University of Technology (AUT), Harry Koprivcic and Student Learning Services and staff for their assistance with statistical analysis.

To my wife Debbie and my daughters Courtney and Laura for all their encouragement and support and for the sacrifices they have made so that I could complete my thesis.

Finally, to Dr Dennis Viehland for encouraging me to enrol in this challenging, informative and rewarding MIS degree.

Table of Contents

List of Tables and Figures

1. INTRODUCTION

1.1 The Importance of Spreadsheet Error Detection

Spreadsheets are one of the most commonly used and popular software products available and are used for a wide variety of purposes due to the software's ease of use and flexibility. Unfortunately the majority of end-user developers have not received any formal training in analysis or development methodologies so the quality of spreadsheets tends to be poor (Ayalew, Clermont & Mittermeir, 2000; Kreie, Cronan, Pendley & Renwick, 2000).

The poor quality of spreadsheet development has led to many decisions being made based on faulty or misleading information (Panko, 2000a, 2000b; Panko & Halverson, 1996). In September 2001, the National Australia Bank had to write off $6.5 billion dollars of its US mortgage business due to an incorrect interest rate being used in a financial model. In June 2003, TransAlta Corporation in Canada was forced to reduce its annual earnings by $24 million due to a spreadsheet error and in July 2004, NASA acknowledged that a faulty spreadsheet model had been used to evaluate the impact of foam striking the wing of the Columbia shuttle (BBC, 2004; EuSpRIG, 2004). These types of errors have served to confirm researchers' warnings that spreadsheets are inherently error-prone and that 30% of the spreadsheets in commercial use are likely to contain undetected errors (Butler, 2002).

1.2 Justification of Research

Studies by Panko (2000a, 2000b), Galletta, Hartzel, Johnson, Joseph and Rustagi (1996) have confirmed the high incidence of errors in spreadsheets and the difficulty human subjects have finding and preventing them.

In 2001, David Nixon and Mike O'Hara conducted a study in which they tested a number of spreadsheet auditing tools against a spreadsheet seeded with errors to determine how well the auditing tools performed. They concluded that the spreadsheet auditing tools were quite useful but did not handle omission errors very well (Nixon & O'Hara, 2001).

While spreadsheet auditing tools may have been inadequate in the past, the use of object technology to develop modern spreadsheet programs means that the auditing programs will have access to a lot more information than would have been available in the days of Lotus 1-2-3 for DOS.

Since some of the spreadsheet information available to the auditing software would not be available, or obvious, to a human reviewer, it is possible that modern spreadsheet auditing programs will perform better than their predecessors did. If this is the case, the combination of human and automated error detection may be the best solution. The ability of the auditing tools to identify obscure or hidden problems and inconsistent formula use, combined with the human ability to comprehend the intent of the spreadsheet, may well result in greater numbers of errors being detected.

By identifying the particular strengths of human reviewers and spreadsheet auditing software, this research may help to clarify the role that audit software should assume and possibly contribute to the refinement of existing audit software.

1.3 Research Objectives

The objective of this study was to examine the ability of human participants and spreadsheet-auditing software to locate spreadsheet errors by having the two groups try to locate errors embedded in a sample spreadsheet. If the study shows that the two groups do find different errors, then it follows that they possess different abilities and so helps to affirm the role of spreadsheet audit tools as an aid to finding spreadsheet errors.

The research aims to show that there is a difference between the errors found by participants and the errors found by spreadsheet audit tools with the measure of success being a statistically significant difference in these two groups of errors.

1.4 Limitations of the Study

The spreadsheet used for this study was modelled after a spreadsheet used in a study by Galletta, Hartzel, Johnson, Joseph & Rustagi, (1997) where 113 MBA students were asked to locate eight errors in a spreadsheet presented on screens and sheets of paper. However, in the 1997 study the students were all present in one place at the

same time whereas in this study it was not possible to get the participants together in a single location so a web-based experiment was selected instead.

The marking of the experiment was complicated by the need to find a marking system that would operate for both participants and spreadsheet audit tools. The intention was to use the marking scheme devised by Nixon and O'Hara (2001); however, it was necessary to create a derivative so that the participants and audit tools could be marked on a similar basis.

1.5 Organisation of the Thesis

After introducing its objectives, the thesis looks at the history of end user computing and the desire of end users for independence, from the early days of computers in the 1940's, through to the mainframe and time-sharing years of the 1960s and the personal computer era in the 1980's. The literature review then traces the history of spreadsheets, the people and companies that contributed to the development of modern spreadsheets such as Lotus 1-2-3 and Microsoft Excel and the consequences of the widespread use and popularity of spreadsheets. Finally, the literature review looks at the problem of spreadsheet errors and the techniques that have been used to try to deal with them.

The research section of the thesis introduces the research hypothesis and describes the experiments and procedures used to collect the results. Some of the demographic data is examined for significance and then the results of the participant and audit tool experiments are tested for statistical significance using ANOVA. The results are then described and summarised.

In the discussion and conclusions section the results of the two experiments and the analysis are discussed, the limitations of the research are explored and suggestions for further research are made.

2 LITERATURE REVIEW

2.1 Introduction

Since the early days of end user computing, concern has been expressed at the lack of formal procedures governing end user development and the potential risks arising from applications that have been poorly designed and developed (Chadwick, 2003; Panko & Halverson, 1996). With the advent of spreadsheets, those concerns were realized when end users were given a programming paradigm that is both friendly and highly versatile, but also one where it was very easy to make mistakes (Freeman, 1996; Godfrey, 1995; Panko & Halverson, 1994).

This chapter traces the history of end user computing, from the early days of the mainframe through to the introduction of spreadsheets. It then reviews the literature detailing the frequency, types and causes of spreadsheet errors and the methods, tools and techniques that have been developed to improve spreadsheet quality.

2.2 End User Computing

2.2.1 The Mainframe Era

The late 1940's saw the birth of the commercial computer. The first was the BINAC built by the Eckert-Mauchly Computer Corporation and financed by American Totalisator. However, in 1949, the Eckert-Mauchly Computer Corporation was sold to Remington-Rand, after American Totalisator withdrew its financial support. Remington-Rand then produced the UNIVAC, a product originally developed by Eckert-Mauchly but incomplete at the time of the takeover. Not to be outdone by the UNIVAC, IBM started producing their model 701 Scientific Computer, in December 1952, followed by the model 702, in September 1953 (Campbell-Kelly & Aspray, 1996; Ceruzzi, 2003). The age of commercial computing had arrived and with it came a new concept – *end user computing*.

As the mainframe market continued to grow, clients of the computer suppliers started to form themselves into user groups. IBM's customers formed a scientific applications group; (SHARE) and a commercial applications group; (GUIDE). They came into existence in 1955 with the objective of sharing user developed

applications and experiences and to try to influence the direction of the IBM Corporation. Similar user groups also formed around the other mainframe suppliers, such as Digital Equipment Corporation's DECUS user group and Univac's USE user group. The suppliers were happy for these groups to exist, as they provided a support structure for their users, were helpful in setting the direction for new technology and positively impacted the growth of the mainframe market (Campbell-Kelly & Aspray, 1996; Ceruzzi, 2003).

The mainframe also changed the way that many companies functioned. In contrast to accounting machines, which could be placed in any room and operated by clerical staff, a typical mainframe site required an air-conditioned environment filled with card readers, tape units, disk drives, printers and a team of highly skilled information technology (IT) professionals. To run a job on the mainframe, a deck of punch cards or a roll of perforated paper tape had to be created and sent to the IT department. The cards would then be run through the mainframe by a computer operator some hours or days later and the results returned to the sender in the form of a printed text on perforated fan-fold paper (Campbell-Kelly & Aspray, 1996). This change to mainframe "batch mode" processing made the IT department a powerful and indispensable player in many businesses.

As IT was the source of technical expertise and it held a monopoly over an organisation's information resource, it was able to put in place procedures and structures to which end users had to adhere in order to gain access to the information they wanted. End users were also required to deal with IT to get new applications developed, special tasks processed and technology purchases authorised (Markus & Bjørn-Andersen, 1987). The dominance by IT was unacceptable to end users who felt that IT was arrogant and unresponsive to business needs, unable to deliver on their promises and generally a barrier to end users doing their jobs.

An ideal situation for end users would be one where the end user could take care of their information technology needs themselves without the need for information gatekeepers or technical experts (Kettinger & Lee, 2002; Luftman, Papp & Brier, 2002; Markus & Bjørn-Andersen, 1987).

Another problem facing end users was the inability to share mainframe time with other users. To complete a task, it was necessary for a user to schedule time on the mainframe, submit their task for processing, retrieve the results, make corrections and, if time permitted, repeat the exercise. Hopefully, by the end of the process, they have something to show for their efforts. The problem with this situation was that the mainframe became, effectively, the end user's personal computer for the period of the booking, even though it was idle for most of that time (Campbell-Kelly & Aspray, 1996). Instead of having to schedule time, end users needed a way where they could do their work when and where they wanted without the delays of batch processing.

2.2.2 Time-sharing

A potential solution to the problem of efficiently managing the mainframe resource came with the introduction of time-sharing systems. The first time-sharing system, called CTSS (Compatible Time-Sharing System), was demonstrated by the Massachusetts Institute of Technology (MIT), in November 1961. The system could be used by three people concurrently and gave each of them the illusion that they had exclusive use of the mainframe. In reality, the mainframe used the interruptions and delays that occurred naturally in each task, to switch rapidly between them and so give the appearance that all the tasks were running simultaneously. After the release of CTSS other systems came into existence in quick succession. The Dartmouth Time Sharing System (DTSS) in 1964, MIT's 160 user Project MAC in 1965, IBM's Model 67 Time Sharing System (TSS) in 1965 and the 1000 user Multics system jointly developed by General Electric, MIT and Bell Laboratories in 1969. Unfortunately the suppliers of time-sharing systems had seriously underestimated the complexity of producing a high-activity multi-user system with the consequence that by the late 1960's it was evident that most time-sharing systems could only support thirty to fifty concurrent users (Campbell-Kelly & Aspray, 1996; Ceruzzi, 2003).

2.2.3 Minicomputers

In the mid-1960's a PDP-8 minicomputer was introduced by Digital Equipment Corporation (DEC) and sold for US$18,000, less than a tenth of the price of a mainframe. The PDP-8 was built using integrated circuitry and was aimed at the

scientific and engineering community who DEC believed would prefer a cheap computer to one that came with an extensive software range and costly peripherals. The machine quickly gained popularity amongst universities and research institutes and in niche areas such as process automation and instrument control. DEC released their PDP-10 machine in 1967. It arrived preloaded with time-sharing software and was so popular that DEC became the dominant supplier of university computer systems. Although they had started out by selling cheap PDP-8's, DEC's PDP-11 computers were able to fetch up to one and a half million dollars by the early 1970's.

The 1970's saw the introduction of a competitor to DEC that was a spin off from the large scale time-sharing fiasco of the 1960's. After the failure of Multics, Bell Laboratories started their own UNIX development project in 1969 under the control of Ken Thompson and Dennis M. Ritchie. Thompson and Ritchie adopted a minimalist, small-is-beautiful philosophy that eventually produced a system capable of running on an obsolete DEC PDP-7. Thompson and Ritchie then convinced the Bell Laboratories patent division to fund a redevelopment of UNIX using "C", a language specifically developed by Ritchie for writing operating systems. The system went into production on a PDP-11/45 and was used to run a text-processing system for preparing patent specifications. Bell Laboratories generous licensing model led to UNIX being adopted by universities and research institutes with the consequence that by the mid-1980's over 250,000 computers were running UNIX operating systems (Campbell-Kelly & Aspray, 1996; Ceruzzi, 2003).

With time-sharing being available on minicomputers, it was only a matter of time before IBM provided the same facility on their mainframes. While time-sharing was impractical on their 360 series of mainframes, it was possible on their new System/370 series so in 1971 IBM made CMS (Conversational Monitor System) and TSO (Time Sharing Option) available to their customers. IBM also released their CICS (Customer Information Control Systems) software, a transaction processing system that allowed customers to develop their own online applications (Ceruzzi, 2003).

2.2.4 End user Development

Despite the advances in operating systems and hardware, applications development software on minicomputers and mainframes failed to keep pace and by the mid-1970's 90 percent of the applications being developed world-wide were still being written using FORTRAN or COBOL (Campbell-Kelly & Aspray, 1996). This situation was unsatisfactory for end users who had anticipated the introduction of user-friendly programming languages only to find themselves still reliant on the IT sector to provide the information they needed (McLean, 1979). IBM tried to address the problem in two ways: firstly, by drawing on the best features of COBOL, FORTRAN and ALGOL to produce PL/1 (Programming Language One) and secondly by releasing their APL (A Programming Language) product. Despite the initial interest in PL/1 due to its extensive features and versatility, the product was regarded as being far too complex and so failed to gain wide acceptance (Ceruzzi, 2003). Conversely, APL received a negative response as it used an odd character set and was more of a notation than a real programming language. However, the interactive, conversational style of APL fared better than PL/1 and with the help of some APL add-on products to facilitate data manipulation, report generation and financial planning, the language did manage to gain a following amongst the end user community (Hammond, 1982; Sammet, 1981). Even though applications development software had lagged behind the advances in hardware and operating systems development, end user computing had finally arrived in the form of APL and end users had the thing they had been longing for - a means of analysing and manipulating data, without having to involve IT personnel.

By the late 1970's and early 1980's, end users were seeing the realisation of their dream to handle their own IT requirements. BASIC, GIS, RPG and MARK IV had been added to the list of mainframe query and report writing tools and some high level languages more suited to end user needs were available including EASYTRIEVE, FOCUS and RAMIS (Benson, 1983).

2.2.5 The move to the PC

Early in 1975 the first computer to use microprocessors appeared in the marketplace in the form of the Altair 8800. The Altair 8800 consisted of a panel with switches and neon bulbs that had to be assembled from an electronics hobby kit that could be purchased for under $400. The machine was not very useful as it did not have a screen or teletype to display output, it did not have a keyboard and only had 256 bytes of memory (Campbell-Kelly & Aspray, 1996; Ceruzzi, 2003).

Unaware of the Altair 8800, Steve Jobs and Stephen Wozniak developed a computer based on the Mostek 6502 processor which they called the *Apple*. The first Apple computer also lacked a keyboard and screen but Jobs and Wozniak still managed to sell two hundred hand-assembled units by 1976.

In 1977, three microprocessor-based machines were produced. Radio Shack released their *TRS-80* which used the Z-80 chip, Commodore released their *PET* computer which used Intel's 8088 chip and Jobs and Wozniak released their *Apple II* computer based on the Mostek 6502. The Apple II used the same chip as the Apple I but the machine came packaged in a clean plastic case, was faster than the TRS-80 and the Commodore PET computers and had colour graphics capabilities. The machine was initially shipped with a version of BASIC written by Stephen Wozniak, but in later machines it was replaced by a version developed for the 6502 by a small software company called Microsoft (Campbell-Kelly & Aspray, 1996; Ceruzzi, 2003).

In August 1981, IBM announced that they were entering the microcomputer market with the *IBM Personal Computer* (PC). The machine was based on the Intel 8088 chip and was shipped with PC-DOS and BASIC software developed by Microsoft (Ceruzzi, 2003). The entrance of IBM into the personal computer arena legitimised the PC for the corporate market and by 1983 the personal computer had become an industry standard and was commonly used to run spreadsheets, word processing and analysis software (Benson, 1983; Campbell-Kelly & Aspray, 1996).

The development of the PC was very significant for end users as it meant they were no longer dependent on an overburdened IT department that usually failed to deliver either on time or on budget. Instead they were able to perform tasks on their own PC's whenever they wanted, in a fraction of the time it had taken previously and best

of all, they had total control over their own information technology needs (Benson, 1983; Rothi & Yen, 1989).

Some companies struggled to deal with the end user computing phenomenon. They recognised that IT had a reputation for delivering projects late and over budget and that there was conflict between IT and end users but allowing unrestrained growth in end user computing was creating its own problems (Rothi & Yen, 1989). To address these concerns, companies adopted a number of different strategies to try to manage their end user computing.

Alavi, Nelson & Weiss (1987) and Rothi & Yen (1989) categorized these different strategies as: laissez-faire, monopolist, accelerated, marketing and operations-based.

Laissez-faire

The *laissez-faire* approach allowed each user to buy whatever they desired and develop whatever they wanted without the need to consult with their IT department. This risked wasting huge amounts of money and ending up with duplicated or incompatible systems but had the short-term benefit of keeping end users happy.

Monopolist

The *monopolist* approach was used by companies who sought to maintain centralised control over technology often under the management of their IT department. While this style of management did address some of the problems created by end user computing, many users saw it as being interference that prevented them doing their jobs properly so they would simply bypass the restrictions.

Accelerated

The *accelerated* strategy encouraged end user computing by providing training, support and technical advice appeared to be the best option to make end users happy. However, by supporting end user computing it also had the effect of inflating the demand for IT resources and consequently making it difficult to provide the support and finance to sustain the growth.

Marketing

The *marketing* model encouraged end users to follow the guidelines and directions of a central group but also gave them access to local support groups if they needed them. The local group also would serve to market the aims of the central group to their users.

Operations-based

The *operations-based* strategy was similar to the monopolist approach in that resources were centrally managed but the focus in an operations-based strategy is to benefit all end users by efficiently utilising the available resources and providing local support and training.

In addition to management, the other area of concern was the poor quality of end user developed software. Of particular concern was the lack of documentation, inadequate backups and integrity problems (Benson, 1983). Some people felt that allowing end users to develop software simply resulted in unstable applications that satisfied the needs of individuals, whilst ignoring or subverting the needs of the company as a whole (Davis, 1989). End user applications were also regarded as being high risk simply because they were developed by untrained and inexperienced people and were the target of frequent, undocumented changes (Alavi, Nelson & Weiss, 1985). Ideally, the strategy chosen needed to reflect the objectives of the company and work towards minimizing the adverse impacts of end user computing. However, one thing was certain; end user computing was here to stay.

2.2.6 Spreadsheets

The application generally credited with driving the growth of end user computing on PC's was VisiCalc (an abbreviation of the words "visible calculator"). VisiCalc, a computerised spreadsheet, was developed in the spring of 1978 by Dan Bricklin whilst studying for his MBA at Harvard Business School (Mamis, 1999; Nelson, 1993; Rose, 1989) and predominantly written by his friend Bob Frankston.

In 1979, Bricklin and Frankston formed the company *Software Arts, Inc*. and entered into an agreement with *Personal Software* (later VisiCorp) to market and distribute VisiCalc (Brandellf, 1999; Ceruzzi, 2003). The program was released in October 1979 for the Apple II and was an instant success. Even though Apple Computers

downplayed the role of VisiCalc in Apple sales, they acknowledged that 25,000 of the 130,000 Apple computers sold before September 1980 were directly attributable to the Apple II being able to run VisiCalc (Campbell-Kelly & Aspray, 1996; Ceruzzi, 2003).

In 1981 the first IBM PC's were produced and by 1983 had become an industry standard. The machine came preloaded with a PC-DOS operating system, a BASIC interpreter supplied by Microsoft and an option to purchase a copy of VisiCalc specifically developed for the IBM PC. Truly, VisiCalc had shown itself to be the first *killer app* – a program so useful that people would be willing to buy a computer simply because it could run VisiCalc.

Unfortunately VisiCorp failed to capitalise on the market leadership gained through VisiCalc and instead got distracted by a damaging court battle with Software Arts that eventually led to VisiCorp's demise (Saffo, 1989). The product which displaced VisiCalc was created by Mitch Kapor, a product manager at Personal Software and the developer of the Visiplot and Visitrend add-on products (Power, 2002). In 1982, Kapor sold the rights to Visiplot and Visitrend for $1.7 million and along with some venture capital, formed the *Lotus Development Company*. Kapor's company developed their new LOTUS 1-2-3 spreadsheet at a cost of $1 million and then spent an estimated $2.5 marketing the product. The strategy was so successful that when the product was finally released in 1983 it sold 60,000 copies in the first month, 850,000 copies within 18 months and by 1985 was the clear leader in the spreadsheet market. The final indignity for VisiCalc came in 1985 when the assets of Software Arts were purchased by Lotus and VisiCalc was removed from the market on the basis that it was an inferior product to Lotus 1-2-3 (Campbell-Kelly & Aspray, 1996; Power, 2002).

Having wrestled control of the spreadsheet market from VisiCalc, Lotus strengthened its position by shipping their Lotus 1-2-3 Release 2.0 spreadsheet and set in place a strategy of acquisition and expansion. In 1986 Lotus celebrated its 2 millionth sale of Lotus 1-2-3 software, followed by 3 million in 1987 and 4 million in 1988. Lotus's acquisition strategy continued through to 1992 and the company expanded into cross-platform development, the home-office market and groupware (Computing Canada, 1992). Then in 1993 Lotus's fortunes changed dramatically. By focussing on

groupware products, the company had been too slow to convert their spreadsheet to the new Windows operating system introduced by Microsoft in 1987.

When Windows 3.0 was shipped in 1989 the only spreadsheet available to run on it was Microsoft Excel. The situation remained that way until 1992 when other spreadsheets, including Lotus, shipped their Windows versions but by that stage Microsoft Excel had gained an unassailable lead on the Windows platform and was the new leader in the spreadsheet market (Power, 2002).

2.3 Spreadsheet Errors

2.3.1 Introduction

As a result of the events that drove end user computing, end users found themselves holding a versatile tool to assist them in making important business decisions and, free of the constraints of a domineering IT department, they were well placed to develop whatever they wished (Cragg & King, 1993). Unfortunately, the situation was not as good as it first appeared. End user disregard for formal development and testing methods combined with the easy-to-use spreadsheet paradigm created a situation where critical decisions were being formulated using unreliable and misleading information (Miric, 1999).

While there was anecdotal evidence of the problem of spreadsheet errors, research conducted since 1987 has confirmed the suspicions that spreadsheet errors are not only commonplace, their presence should be assumed (Panko, 1998). Of particular concern was a 1997 field audit of spreadsheets conducted by KPMG Management Consulting in London and summarised in KPMG South Africa's article "The hidden risks of spreadsheets and End User Computing", in which they found that 95% of the spreadsheets reviewed had significant errors, 92% of the tax related spreadsheets made taxation errors, 75% of the spreadsheets had accounting errors and 59% were poorly designed (Miric, 1999).

Considering the importance of spreadsheets in decision support and the extensive use of spreadsheets in business, it is important to understand why spreadsheets are so popular, why they have such a high incidence of errors and what can be done to prevent and detect them.

2.3.2 Why use spreadsheets?

There are many reasons why spreadsheets are so popular, but five factors seem to dominate: i) spreadsheets are easy to use, ii) the row and column style is well understood (Mason, 1989), iii) it is an easy programming language to learn and iv) domain knowledge is maintained (Nardi & Miller, 1990a).

2.3.2.1 Spreadsheets are easy to use:

The modular design and flexibility of spreadsheets means that it is possible to use them with a minimal amount of training and then use that knowledge to perform a multitude of tasks. The rapid feedback and prompting from the spreadsheet environment helps the end user to successfully enter spreadsheet functions making the user feel empowered and capable (Horowitz, 2004; Mason, 1989; Moström & Carr, 1997)..

Spreadsheets have been described as being the *Swiss Army knife* of software and the original *agile software development environment*. This is because you can do almost anything with them from creating contact lists, to accounting, to database tasks. So it is natural that if a problem presents itself that users will reach for a spreadsheet to solve the problem, even if it is not the best tool to use (Mason, 1989; O'Beirne, 2003; Ridington, 1988).

2.3.2.2 The row and column style is well understood:

The tabular structure of spreadsheets is a well-understood and natural construct that makes it easy to navigate around the spreadsheet and to enter and view information. This is because people have been accustomed to reading information from tables such as calendars and multiplication tables since they were very young and the combination of rows, columns and labels makes spreadsheets easy to comprehend and to memorise (Moström & Carr, 1997; Nardi & Miller, 1990b).

The benefit of using tables was well understood by the developers of VisiCalc who modelled their spreadsheet after the columnar paper used by accountants. This simple structure allowed people to become productive quickly without requiring much thought (Mason, 1989; Nardi & Miller, 1990b).

2.3.2.3 It is an easy programming language to learn:

Spreadsheets use a function based programming language that makes it very easy for users to enter the correct formulae (Nardi & Miller, 1990b) and knowing even a modest number of commands and functions can be sufficient to make a user productive (Moström & Carr, 1997). This ease of usage means that many people are not even aware that they are performing a task very similar to programming.

2.3.2.4 Domain knowledge is maintained

Some end users prefer not to impart domain knowledge to others so that the recipient can then turn the knowledge into a program. This reluctance is particularly strong if the recipient of that knowledge is from an IT background. The benefit of spreadsheets is that they allow an end user to build one by themselves without having to impart the knowledge to anyone (Clermont, 2003a; Cragg & King, 1993; Nardi & Miller, 1990a).

Spreadsheets provide end users with a simple programming language they can learn part or all of, to turn their own ideas into workable models using a familiar development environment. It is not surprising, then, that spreadsheets are such a popular tool, but with ease of use also comes the possibility of making mistakes.

2.3.3 Types of Spreadsheet Errors

Spreadsheet errors are a widespread problem and the reasons for the errors are numerous. Rajalingham, Chadwick and Knight (2000) defined a classification system for spreadsheet errors in which they identified two major classes of errors; specifically *system generated* errors and *user generated* errors. User generated errors were further classified into *quantitative accidental, quantitative reasoning, qualitative semantic* and *qualitative maintainability* errors (Rajalingham et al., 2000). Each of these error types is examined below.

2.3.3.1 System Generated Errors:

System generated errors are the result of the spreadsheet application producing incorrect results either deliberately or as a result of a software bug. Some of these anomalies appear in Microsoft's Excel spreadsheet and are shown below:

Y2K Errors

Entering the date 18/06/00 (without the century) into Microsoft Excel produces the date 18/06/2000. However entering the same digits using the DATE function (i.e. "=DATE(00,06,18)") produces the date 18/06/1900. This could produce incorrect results but can be easily circumvented by entering the digits for the century (Microsoft, 2003).

Leap Years

Excel also contains a deliberate error related to leap years. As 1900 is not a leap year, Excel should flag the date 29/02/1900 as being incorrect or convert it to 01/03/1900. However Microsoft chose to allow Excel to accept the date as being valid by deliberately duplicating a bug found in early versions of Lotus 1-2-3 which incorrectly calculated 1900 as being a leap year (Microsoft, 2003).

TRUE and FALSE

In Microsoft Excel, TRUE has the value 1 and FALSE the value 0. This can be confirmed by entering the formula "=SUM(TRUE,TRUE,FALSE)" into a cell and checking that it returns the value 2. However entering the value TRUE into cell A1, TRUE into cell A2 and FALSE into cell A3 and then using the formula "=SUM(A1:A3)" to total them produces a result of 0 (Walkenbach, 2004). Entering the formula "=IF(TRUE=1,"YES","NO")" into a cell makes the situation even more confusing by producing the result "NO" which suggests that TRUE does not equal "1" after all.

As might be expected, system generated errors are not very common and just finding descriptions of the three problems listed above was difficult. Much more common are user-generated errors, which make up the bulk of the errors affecting spreadsheets.

2.3.3.2 User Generated Errors:

User generated errors are those errors which are caused by the user of the spreadsheet and not the software. Studies into human error rates confirm that when people are required to do mundane tasks, 0.5% of the activities they perform will

result in errors and the rate increases to 5% as the tasks become more difficult (Panko, 1997a, 2000a).

There are two categories of user-generated errors, *quantitative* and *qualitative* (Panko & Halverson, 1996; Rajalingham et al., 2000). Quantitative errors are those errors which produce incorrect results in the spreadsheet and are made up of *accidental* and *reasoning* errors. Qualitative errors are usually the result of poor spreadsheet design and may make the spreadsheet less useable or lead to the results of the spreadsheet being misinterpreted. Qualitative errors comprise *semantic* and *maintenance* errors (Rajalingham et al., 2000).

Quantitative Accidental Errors

Accidental errors are those errors caused by carelessness or inattention; such as typing errors. These errors are usually noticed by the developer and corrected early. However, if any do get through undetected, they are likely to result in the spreadsheet producing incorrect output (Rajalingham et al., 2000). Accidental errors fit into one of three categories: *omission* errors, *alteration* errors and *duplication* errors.

Omission errors are those where an important piece of information, value or formula are left out of the spreadsheet. Research into human factors has shown that these errors are very hard errors to detect (Panko & Halverson, 1996; Rajalingham et al., 2000) and they are often discovered by someone other than the developer (Ross, 1996).

Alteration errors occur when a change is made that introduces a fault into the spreadsheet. The fault could be something as simple as protecting an input cell so that it can't be used, sorting a group of cells but leaving a required column out of the sort or overtyping a formula with the value the formula produces. These errors are also difficult to detect as the impact they have on the spreadsheet may not be immediately obvious (Panko & Halverson, 1996; Rajalingham et al., 2000).

Duplication errors occur where the same information is introduced into the spreadsheet more than once. An example of this kind of error could be an inflation rate being applied twice in the spreadsheet or a row being copied and then included twice in a total (Rajalingham et al., 2000).

Quantitative Reasoning Errors

Reasoning errors are those which are introduced when the wrong formula has been selected to perform a task or the developer tries to solve a problem in the wrong way through having insufficient knowledge. Reasoning errors are usually either *implementation* or *domain knowledge errors* (Rajalingham et al., 2000).

Implementation errors are the consequence of the developer using the functions and features of the spreadsheet incorrectly so that syntax or logic errors are produced. Syntax errors are easy to identify, as they will be highlighted by the spreadsheet but logic errors are the incorrect use of a formula and so are much harder for either the spreadsheet or the developer to detect (Rajalingham et al., 2000).

Some common logic errors are: a) Circular references, where the formula uses the cell it is in or another cell that refers to it in its formula. b) Referencing errors, where a cell is copied and the references in the formula point to new cells relative to the new location instead of the original location as was intended and c) Formula errors, where a formula is syntactically correct but has been used incorrectly. An example of a formula error is the formula "=AVERAGE(SUM(A1:A10),SUM(B1:B10))" where the answer will be the average of the total of the cells A1 to A10 and the total of the cells B1 to B10, instead of being the average of all the cells in the range A1-B10 (Rajalingham et al., 2000).

Domain knowledge errors occur when the developer does not understand a situation well enough to produce the correct code in the spreadsheet. Domain errors also come in two varieties: *Real world knowledge* and *mathematical representation* errors. Real-world-knowledge errors occur when the wrong formula is used to derive a result, such as dividing a total number of days by 365 to determine the number of years but disregarding the impact of leap years. Mathematical representation errors occur when a formula is written incorrectly so that it does not perform the intended function. Operator precedence errors are one example of a mathematical representation error (Rajalingham et al., 2000).

Qualitative Semantic Errors

Semantic errors come in two forms: *structural* and *temporal* errors. They have the effect of distorting what the spreadsheet is trying to represent and may lead to poor decisions being made. Structural errors are problems with the layout, appearance or formatting of the spreadsheet. Examples of this problem are where titles or labels are incorrect, the cell formatting does not accurately represent the values in the cells or a formula is entered that will produce the right result but has not been used correctly. Temporal errors are caused by using out-of-date data. They can occur when an external data source has been referenced that has not been refreshed or a value has been entered that has not been kept up to date (e.g. exchange rates) (Rajalingham et al., 2000).

Qualitative Maintenance Errors

Maintenance errors are those attributes of the spreadsheet that make it difficult to preserve quality and reliability. While some of them may not be errors as such, they may lead to errors over the lifetime of the spreadsheet. Hard coded values, matrix operations and excessively complex formulae are three examples of this problem (Rajalingham et al., 2000).

Spreadsheet software has shown itself to be very easy to use both correctly and incorrectly. With the many varieties of spreadsheet errors that are possible, it is understandable that Panko and Halverson would find that 38% to 77% of “finished” spreadsheets contained errors (Panko & Halverson, 1996).

2.3.4 Dealing with spreadsheet errors

Recognising the problems that spreadsheet errors can cause, the academic and commercial sectors have tried to address the problem in a number of different ways. In this section the techniques used to deal with spreadsheet errors are examined including: recognising the problem, development methodologies, peer involvement, development aids and spreadsheet auditing tools.

2.3.4.1 Recognising the problem

Before spreadsheet errors can be dealt with, it is necessary to recognise that there is actually a problem. Since spreadsheet quality is often evaluated by the person or

persons involved in its development, they are inclined to view the spreadsheet more positively than if it had been developed by someone else (McGill et al., 2000). People also mistakenly believe that information generated “by the computer” is valid (Ahmad, Antoniu, Goldwater & Krishnamurthi, 2003; Seymour, 1984) and are confident that any spreadsheets they produce are error free (Brown & Gould, 1987).

Getting people to recognise the threat presented by spreadsheets is getting easier as each new spreadsheet horror story appears in the press (EuSpRIG, 2004). However, if end users remained unconvinced of the threat of spreadsheets errors, an audit of their most important spreadsheets should remove any doubt (Panko, 1995a, 1997b; Ross, 1996). Once the problem with spreadsheets is acknowledged, it is easier to put steps in place to address the issues that cause them.

2.3.4.2 Development methodologies

As the problem of spreadsheet errors has become more apparent, research papers, journal articles and web sites have appeared promoting best practice techniques and methodologies for developing and managing spreadsheets. It is proposed that if end users could be convinced to use a systems development life cycle (SDLC) to develop their spreadsheets, the quality of the spreadsheets would improve and ongoing maintenance problems would be reduced (Read & Batson, 1999).

The difficulty with introducing formal methodologies into spreadsheet development is deciding who should control them and how stringently they should be applied. This becomes especially difficult when some users refuse to accept that spreadsheets are programs and so should not be subject to formal controls and attempting to introduce them will return end user computing to the early days of IT domination (Ayalew et al., 2000; Ronen, Palley & Lucas, 1989; Stone et al., 1989).

The solution may be to provide training to end users to show them the benefits of using a more formal approach employing some simple techniques, whilst limiting the impact of the process. One such methodology was defined by Read and Batson (1999)based around a six stage software development life cycle consisting of scope, specify, design, build, test and use.

Scope

In the scope stage the user decides what they will include in the spreadsheet, the assumptions that will be used, the people and time resource required and how complex it will be. This is also the stage at which significant stakeholders should be consulted to ensure that they happy with the direction of the project (Read & Batson, 1999).

Specify

In the specify stage, the purpose of the spreadsheet is carefully considered and the calculations and logic required to build the spreadsheet are defined in sufficient detail that they can be clearly understood (Read & Batson, 1999).

Design

In the design stage a spreadsheet structure is produced with the objective of making the spreadsheet easy to use, easy to maintain and likely to reduce errors (Read & Batson, 1999). This stage is very important as research has shown that the majority of spreadsheets contain errors that could have been prevented during the design stage but because they were missed, are likely to lead to problems in the future (Panko, 1995b).

A technique used to make spreadsheets easier to follow is to design them so that they read from the front sheet to the back, top to bottom and left to right so that the spreadsheet has consistent flow. This helps to prevent the introduction of circular references by ensuring that referenced cells are above and to the left of the cells doing the referencing (Blood, 2002; Raffensperger, 2003; Read & Batson, 1999). Since this layout is very similar to the way we read a book, it should feel natural to users and make spreadsheet navigation easier (Read & Batson, 1999)

Another technique is to create *input*, *calculation* and *output* sections and place them in logically separated areas of the spreadsheet (Figure 2.1); either positioned diagonally across the spreadsheet so that they do not have any rows or columns in common or stored on different worksheets.

This arrangement ensures that adding or deleting rows and columns will not impact on other areas of the spreadsheet (Freeman, 1996; Stone et al., 1989)

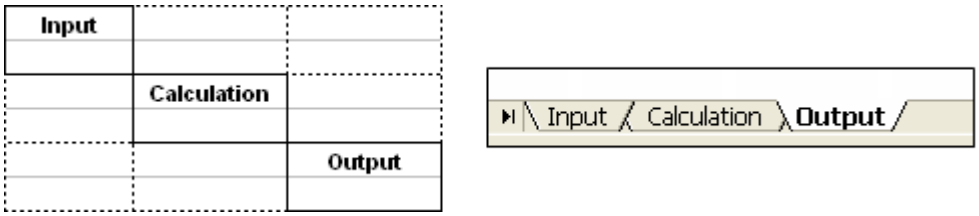


Figure 2.1 Logically Organised Regions

The input area is a *data only area* that does not contain any formulae or calculations and is used to provide the source variables for the rest of the spreadsheet. The variables should be carefully labelled with cell formatting and protection being used to indicate which items are labels and which are variables (Figure 2.2).

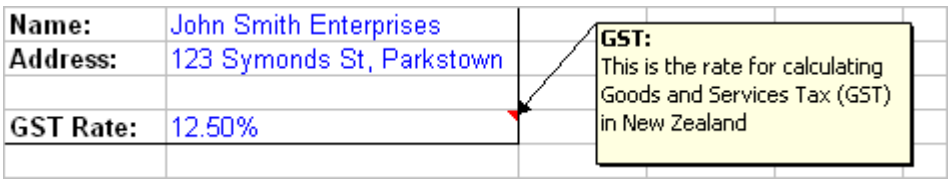


Figure 2.2 Data only area

Adding documentation in the form of text boxes or cell comments will also help clarify what the purpose of each variable is. This has the benefit of centralising the *options* for the spreadsheet and makes it easier for subsequent users to understand what each field is doing and whether they are permitted to change them (Conatser, 1995; Conway & Ragsdale, 1997; Freeman, 1996; Stone et al., 1989).

The calculation area is where most of the calculations take place with the exception of subtotals and totals. This portion of the spreadsheet should reference information in the input area and should not contain its own variables. The calculations should be kept simple, used in a logical sequence, with labels and documentation.

The output (or report) section references the input and calculation sections to produce the output page. The only calculations that should appear in this section are those necessary to produce the subtotals and totals. The output section should be cleanly formatted and include titles and dates as required and, if possible, able to fit on a single sheet (Raffensperger, 2003; Stone et al., 1989).

Build

Before the build stage commences, the developer should confirm that they have the necessary formulae and calculations available to achieve their task and that any assumptions they have about the project are confirmed. The output from the spreadsheet should reflect the needs of the eventual user and be easy read and understand (Read & Batson, 1999; Ronen et al., 1989; Ross, 1996; Stone et al., 1989).

Some of the recommendations that have been made to help build a better spreadsheet include: avoid using complex formulae, use named ranges, add validation and reasonability checks, use cell protection, avoid hard coded constants and Do not hard code constants in formulae (Ettema, Janssen & de Swart, 2001).

Avoid using complex formula

To ensure that the spreadsheet can be maintained, complex formulae, particularly those with several embedded "IF" statements need to be avoided as they tend to cause errors and make the formula hard to understand.

To reduce the risks caused by overly complex formulae the developer should aim to simplify the formula as much as possible by eliminating redundant cell references and operators or by splitting the formula into smaller segments and storing them in separate cells. The formula is probably too complex if the purpose of the formula is not self evident or it uses three or more functions. This issue can make debugging harder by making it difficult to determine intermediate results or to trace precedent and dependant cells (Berglas & Hoare, 1999; Blood, 2002; Freeman, 1996).

Raffensperger (2003) summarised the important aspects of formula simplification in a "*Factors in Cell Readability"* table (Figure 2.3).

Less readable	More readable
Irregular terms reference various ranges. References appear more than once.	Similar terms repeat and reference similar ranges.
References are ordered randomly in the formula.	References read left to right in row and column order.
Well-known formulas are made cryptic by separation into multiple cells.	Well-known formulas (e.g., the quadratic formula) are in a single cell.
Division appears randomly.	Division generally appears at the end.
Formulas contain constants or reference blanks.	Formulas reference constants.
Formulas have unnecessary parentheses and spaces; the formula can be simplified.	Formulas have the fewest characters

Figure 2.3 Factors in Cell Readability Raffensperger (2003)

A key aim in developing spreadsheets in a methodical fashion is to try to ensure that the spreadsheet is constructed in as compact a fashion as possible (Stone et al., 1989). Unfortunately, with the only real limitation on spreadsheet size being available memory, it is easy for end users to build large spreadsheets that increase in complexity along with the size (Mason, 1989). Since a medium-sized spreadsheet uses between 500 and 1,500 distinct formulas (Ettema et al., 2001), it is understandable that as the size grows, so do the number of formulae and the potential for errors. The size will also make errors harder to detect (Simkin, 1987) and any missed errors are likely to impact adversely on the quality and reliability of the spreadsheet (Stone et al., 1989).

Working with large spreadsheets can be difficult but there are two options for making them easier to manage. The first is to break the spreadsheet into segments and store each segment on a separate worksheet in the same or another workbook (Janvrin & Morris, 2000; Stone et al., 1989). The sheets can then be tied together using inter-worksheet and inter-workbook links. The problem with this solution is that the use of links has been shown to increase the incidence of spreadsheet errors (Janvrin & Morris, 2000; Morrison, Morrison, Melrose & Wilson, 2002). The second option is to leave the spreadsheet intact but use tools that make the spreadsheet easier to understand, to maintain and to highlight potential problems automatically (Clermont, Hanin & Mittermeir, 2002; Morrison et al., 2002).

Ultimately, the best option is to follow structured spreadsheet design techniques and keep the spreadsheets compact so that the problems associated with large spreadsheets are avoided.

Use named ranges

Named ranges are often promoted as being the solution to many of the problems plaguing spreadsheets, as they allow the developer to use labels to represent cell locations. This is because named ranges are more memorable, produce a formula that is more intuitive and are easier to understand than cell coordinates (Berglas & Hoare, 1999; Bradley, 2003; Whittle & Janvrin, 2002). In a study of 215 undergraduate students, Whittle & Janvin (2002) found that the use of named ranges improves the quality of the spreadsheet and increases developer satisfaction without affecting development time. Given these benefits it is understandable that their use should be

encouraged. However, what is not so well understood is the problems that named ranges present.

Named ranges do have some inherent problems that mean they must be used with care. The first issue that needs to be clarified is that the titles *named range* or *named cell* are incorrect as they are actually *named formulas*. This is because the named formula does not reside in a cell at all; instead it is created in memory as a reference to a formula, a value or a cell range and then saved along with the spreadsheet. It is, therefore, an important distinction, as it makes it easier to understand why Excel will allow you to assign the name *GST* to the value *12.5%*, even though the value 12.5% does not exist in any cell in the spreadsheet (Walkenbach, 2004). A side effect of this distinction occurs when the cells to which a named formula refers to are deleted. Excel will retain the named formula but it changes the reference to "#REF!" because the target of the reference has been removed (Walkenbach, 1999).

Copying is also a problem with named formulae. When a sheet is copied from one spreadsheet to another, any defined names that do not exist in the receiving spreadsheet are automatically defined by Excel and referenced back to the originating spreadsheet, all without the developer being told. This can produce *ghost links* that the developer is not even aware exist and that can only be removed by deleting the named formulae that were automatically generated by Excel.

Even copies within spreadsheets can be messy if workbook level names are involved. Named formulae can be defined local to a sheet as *worksheet level names* or global to the spreadsheet as *workbook-level names*. If you define a global name *MyCell* that refers to the cell A1 on Sheet1 and then copy the entire sheet to create sheet2, you will find that *MyCell* has been recreated as a workbook-level name on sheet2 that refers to cell A1 on sheet2. This is a problem because when you are on sheet2, the global name *MyCell* that refers to Sheet1!A1 is hidden by the local name *MyCell* that refers to Sheet2!A1. This raises the possibility that if cell sheet1!A1 is ever updated, the formulae on sheet2 will never see the change (Blood, 2002; Walkenbach, 1999).

In summary, despite the general enthusiasm towards named formulae, they do present their own problems and may affect the quality of the spreadsheet. Developers using them need to have a thorough understanding of their impact and consider whether the benefits gained from named formulae justify their use.

Add validation and reasonability checks

Input validation and reasonability checks can also be used to detect spreadsheet errors. Input validation automatically checks the data being entered into a cell and optionally displays a message if the data is incorrect. In Excel, the "Data, Validation" toolbar option provides the input validation function and allows the spreadsheet developer to place limits on the input to a cell so that it must meet specific criteria before the input will be accepted. This can include restricting the input to a particular data type such as numbers, dates or times or limiting the range of values that are permitted. Excel also allows the input to be validated against a list of items which it presents to the user in the form of a drop-down list when the validated cell is selected (Bradley, 2003; Callahan, 2002; Hormann, 1999).

The alternative to performing data validation on input is to perform reasonability and consistency checks after data has been entered (Simkin, 1987). Reasonability checks are similar to post-conditions assertions in other programming languages (Cook et al., 2004) and can be used to test for faulty logic, incorrect data or to validate one set of results against another (Simkin, 1987).

In experiments using assertions with Forms/3, it was found that assertions had three significant benefits: 1) they made debugging easier and quicker; 2) they could be used to test for a wide variety of spreadsheet errors and 3) they helped the end user developer to understand how easily errors could occur in spreadsheets (Burnett et al., 2003). An example of a simple reasonability check is the use of Excel's "IF" or "ISERROR" functions to test for a divide-by-zero situation before it occurs (Bradley, 2003).

A more complex reasonability check is shown in Figure 2.4 where conditional formatting has been applied to the interest column to notify the spreadsheet user if the interest rate goes outside the expected range of 5% to 10%.

Deposit	Interest	Total
1200.00	8.00%	1296.00
1500.00	1.00%	1515.00
2400.00	7.50%	2580.00
3600.00	12.00%	4032.00
		9423.00

Figure 2.4 Reasonability Checks

Use cell protection and formatting

Shading, borders and protection should be used to highlight various aspects of the spreadsheet (Bradley, 2001) and to indicate those areas that are used for input and those that are used for reporting (Conway & Ragsdale, 1997). Input cells can be marked in a distinct colour and left unprotected, while output cells can be formatted as part of the output page and protected to prevent accidental overtyping of the formulae (Bradley, 2003).

When a report is data intensive and laid out in rows, it is helpful to shade every second row to make reading easier (Figure 2.5), in columnar reports every second column can be shaded and in documents such as invoices the aim should be to make the individual items and totals easy to locate (Bradley, 2001).

Surprisingly, in a study conducted by Cragg and King (1993) only 30% of the spreadsheets they examined used cell protection and only 25% used formatting to make the output easier to read. (Cragg & King, 1993). Fortunately the benefits of spreadsheet formatting and protection are being recognised so the future should see cleaner, more secure spreadsheets being produced (Bradley, 2001; Cragg & King, 1993).

ID	Date/Time	IP Address
303899	10/11/2003 11:00	10.33.237.195
303899	10/11/2003 11:00	10.33.25.61
303899	10/11/2003 11:00	10.33.27.127
365682	07/11/2003 15:00	10.33.215.1
365682	07/11/2003 15:00	10.33.136.230
365682	07/11/2003 15:00	10.33.242.28
349382	05/11/2003 11:00	10.33.110.156

Figure 2.5 Row Formatting

Do not hard code constants in formulae

Using hard coded constants in a spreadsheet formula is not an error but can lead to future maintenance problems. As the constant could appear anywhere in the spreadsheet, it is necessary to search every cell for copies of the constant and to understand how each copy is being used before it can be changed. The alternative to using hard coded constants is to assign the value to a named formula or store the value in a cell that is the target of a named formula. This allows a value such as GST to be defined once and then be used throughout the spreadsheet. Making a change to the GST rate only has to occur in one place for all the dependent formulae to be updated (Berglas & Hoare, 1998, 1999; Butler, 2002; Miric, 1999). Properly handling constants through defining and labelling them will help to make the spreadsheet more maintainable and reliable (Conway & Ragsdale, 1997).

Test

The aim of the testing stage is to locate errors and discrepancies in the spreadsheet and correct them before the spreadsheet goes into production (Read & Batson, 1999). Even though testing is an important aspect of traditional programming and contributes significantly to the quality and reliability of software (Panko, 1999), spreadsheets are usually developed by end users (Ayalew et al., 2000; Stone et al., 1989) and despite the recognition given to high error rates in spreadsheets, they are not usually subjected to the extensive testing applied during traditional software development (Panko, 2000a).

To make development easier a number of techniques have been proposed to improve the quality of end user testing. One technique is to use input data that will produce known results to confirm that the output that is generated meets expectations (Bradley, 2003; Butler, 2002). A version of such testing, called *sensitivity analysis*, involves making a minor change to the value of an input cell to see how it affects the rest of the spreadsheet. Therefore, an increase of one hour on a timesheet should see the salary figure increase by one unit of the workers hourly pay rate (Simkin, 2004). Another technique is to test a SUM() formulae by entering the number ‘1’ into each cell in the SUM formula range and then checking to see if the result is equal to the number of cells that are being totalled (Simkin, 1987). Plotting important data on a graph may also help by displaying a bulge or extra large value in the graph indication that an unusually large number has been entered (Simkin, 1987). Typing incorrect

input data into cells will confirm whether the data is rejected by the spreadsheet and attempting to overwrite formula cells and input cells will verify whether the cell protection settings have been applied correctly (Bradley, 2003). While these techniques are not equivalent to the extensive testing performed by application developers, they should help to eliminate some of the more common errors found in spreadsheets and so improve the spreadsheets quality and reliability.

Use

Two tasks essential to a successful distribution of software are documentation and training. The documentation must clearly explain to the users of the spreadsheet how it functions, any limitations or restrictions it may have, any assumptions that were used to develop the spreadsheet and how it should be used. Providing documentation will also extend the longevity of the spreadsheet by ensuring that it can be understood long after the original developer has moved on. Cell comments can be used to document individual cells if the formula is complex, needs further explanation or where the cell is being used as an input parameter. Some authors suggest including the developer's name, spreadsheet creation date, a description of the spreadsheet logic, a revision history and impact analysis in the spreadsheet, possibly in an *introduction* section or using the "File, properties" dialog in Microsoft Excel (Blood, 2002; Freeman, 1996; Stone et al., 1989).

The impact of using formal development methodologies can be minimised by differentiating between those spreadsheets that will be used by the developer and those that will be used by others. If the spreadsheet is being developed as a one-off activity and unlikely to be used again, it is reasonable that a formal development methodology could be omitted. However, when a spreadsheet is developed for general distribution, it should go through a formal planning and testing procedure as this process has been shown to improve the usability, reliability and auditability of the spreadsheet (Babbitt, Galletta & Lopes, 1998; Davis, 1996; Janvrin & Morris, 2000; Nardi & Miller, 1990a; Rothi & Yen, 1989).

The benefits of applying a structured design methodology to spreadsheets have been widely documented by the academic community (Panko & Halverson, 1996; Rajalingham, Chadwick, Knight & Dilwyn, 2002). However, the lack of a dominant methodology (e.g. Bradley, 2003; Callahan, 2002; Hormann, 1999; Stone et al., 1989) combined with end user antipathy towards the imposition of software development rules have made it difficult for structured design methodologies to gain wide acceptance.

By applying the scope, specify, design, build and test steps defined by Read and Batson (1999), end users can gain the benefits of structured spreadsheet design and reduce the incidence of common spreadsheet errors without having to adopt a formal methodology.

2.3.4.3 Peer involvement

Research has shown that the involvement of peers in the development and review of spreadsheets can improve both the quality and reliability of spreadsheets (Panko & Halverson, 1994).

The traditional view was that spreadsheets were usually developed by individuals. However, one study revealed that the majority of the spreadsheets the study examined had benefited from the participation of two or more people. This situation came about because the spreadsheets acted as a focal point for the informal sharing of spreadsheet information and domain knowledge. The higher skilled users also provided training to their workmates and helped them to construct and debug their spreadsheets (Nardi & Miller, 1990a).

Research by Galletta, et al. (1997) and Panko (1999) confirmed that having individuals or groups examine a spreadsheet for errors is another strategy that will help to reduce them. Panko performed an experiment with thirty third-year and fourth-year graduates in which they were asked to locate errors in a sample spreadsheet. Working alone, the participants located 63% of the errors but when they worked in groups the rate went up to 86%. While the results were higher than those recorded in Galletta's research, this was attributed to the extensions that Panko had added to his study, which were not present in Galletta's. Galletta also found that domain experience helped participants to locate errors and spreadsheet experience

helped improve the speed at which they found them. Presentation style, whether on a screen or on paper, also impacted on the participants ability to locate the errors (Galletta et al., 1993; Galletta et al., 1997).

While these results are encouraging, the inability of participants to find simple spreadsheets errors while self-checking or during peer review shows that these techniques alone will not prevent errors from appearing in spreadsheets.

2.3.4.4 Development aids

A number of development aids and techniques have been created to assist with building, maintaining and testing spreadsheets. The tools fit into three main categories: modelling, and generation tools, testing tools and visualisation tools.

Modelling and Generation Tools

Isakowitz, Schocken and Lucas (1995) identified four properties that exist in all spreadsheets: schema, data, editorial and binding. They developed a series of macros that allowed the spreadsheet user to export the characteristics of the spreadsheet to an external representation and then have the details maintained automatically by the spreadsheet. Whenever a spreadsheet is opened an auto-open macro verifies that the spreadsheet and the exported characteristics still match and if there have been any changes it utilises end user input and regeneration to update the characteristics. They believe that by integrating their methodology into existing spreadsheet programs and storing and managing the external definitions automatically, they can make spreadsheets more portable, easier to understand and easier to maintain (Isakowitz et al., 1995).

A similar methodology was used by Jocelyn Paine for her *Model Master (MM)* programme. Model Master transforms Excel spreadsheets into MM programmes that describe the attributes, logic and formatting of the original spreadsheet. By working with the MM definitions instead of the spreadsheet the end user developer gets the benefits of improved readability, a modular approach to development and the ability to share code between developers and only has to run a simple compile to transform the MM program back into a spreadsheet (Paine, 2004).

Instead of changing the development methodology, the Computer Science department at Oregon State University developed a system in 2002 that allowed the spreadsheet developer to generate test cases that they could use to verify their spreadsheets. These test cases could be built for a portion or all of the spreadsheet and the test data generated randomly or using a goal-oriented approach, depending on the requirements of the user. The benefit of this model is that it simplifies the testing stage of spreadsheet development for the end user and makes the testing process more efficient and comprehensive (Fisher, Cao, Rothermel, Cook & Burnett, 2002).

While these tools are still in the preliminary stages of development, initial observations show that they could be used to improve spreadsheet developer productivity and enhance the quality and maintainability of the spreadsheets. Further research would be needed to confirm whether the additional work involved to use these tools would discourage end users from utilising them.

Testing Tools

Many of the testing tools have been constructed using the Forms/3 spreadsheet research package developed by Margaret Burnett at Oregon State University. Forms/3 has been used to show how complete spreadsheet testing is (Ruthruff, Phalgune, Beckwith, Burnett & Cook, 2004), provide real time debugging information (Reichwein, Rothermel & Burnett, 1999), show how thoroughly homogeneous grids have been tested (Burnett, Sheretov & Rothermel, 1999), provide guards or assertions to validate cells (Beckwith, Burnett & Cook, 2002), suggest values to be used for cell testing (Cook et al., 2004) and to provide visual feedback when testing spreadsheet cells (Rothermel et al., 2000). While these tools will improve the robustness and quality of the spreadsheets with which they are used, they suffer from the limitation that the Forms/3 software can only run on Sun/Solaris and Hewlett-Packard/HP-UX platforms (Burnett, 1996) and so cannot be used with popular commercial spreadsheets like Lotus 1-2-3 and Microsoft Excel (Randolph, Morris & Lee, 2002).

Recognising the value of the *What You See Is What You Test* (WYSIWYT) methodology developed by Rothermel et al (2000) using Forms/3, Randolph, Morris and Lee (2002) developed a version of WYSIWYT for the Microsoft Windows platform using Microsoft's Visual Java (J++), Visual Basic for Applications (VBA) and Component Object Model (COM) technology.

The software is tightly integrated with Lotus 1-2-3 and Microsoft Excel and uses VBA macros to monitor cell activity and adjust spreadsheet cell border colours to let the developer know which cells need further testing (Randolph et al., 2002).

Visualisation tools

Visualisation tools give developers a different view of the spreadsheet by visually portraying the different aspects of the spreadsheet such as cell dependencies, spreadsheet structure, cell equivalence and logic flow.

Microsoft Excel provides a *range finder* tool and an *auditing toolbar* (Figure 2.6) to help users understand what the spreadsheet is doing. The "Range Finder" tool is used whenever a cell is edited in Microsoft Excel.

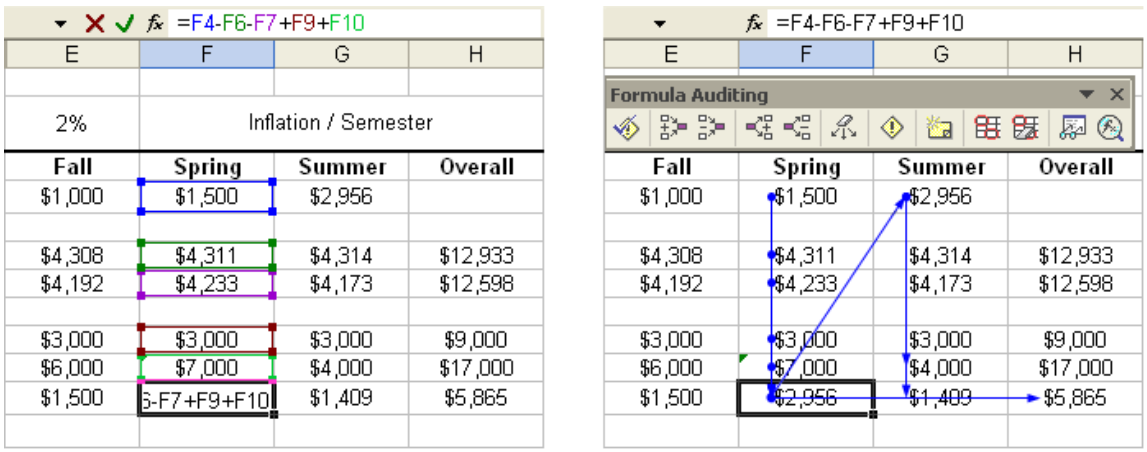


Figure 2.6 Factors Range Finder and Auditing Toolbar

The spreadsheet displays each cell reference in the formula in its own distinct colour and the corresponding cells in the spreadsheet are overlaid with a rectangle using the same colour. This allows the user to determine which cells the formula is using and make corrections, if necessary, by editing the formula or adjusting the size or location of the rectangle.

A difficulty with this style is that it is not always easy to locate the cells which have been highlighted especially when the cells are distant from the formula being examined or a named formula is used and there is no indication where it might reside (Ballinger, Biddle & Noble, 2003b).

The "Auditing Toolbar" can also be used to display arrows that show cell precedents and dependents. The user can then navigate around the referenced cells by double clicking on the arrowheads. The problem with this technique is that the spreadsheet display can quickly become cluttered as additional arrows are added and it is a cell-by-cell process that does not provide any way of getting an overview of the spreadsheet (Ballinger et al., 2003b; Igarashi, Mackinlay, Chang & Zellweger, 1998).

Igarashi et al. observed that the combination of display and programming paradigms in a spreadsheet make it difficult for spreadsheet users to discern the structure and flow of the spreadsheet. To address this problem they developed a set of fluid visualisations that allow the user to navigate around the spreadsheet and gain insight into how it operates. The visualisations provide a *transient local view* (Figure 2.7) that shows the cells which provide the input and receive the output from the selected cell, a *static global view* that shows an overall view of the spreadsheet with the inputs and outputs displayed and an *animated global view* which displays the spreadsheet structure using a series of animations as the user interacts with it. The visualisation tools were developed using Pad++ and Python and can only run on UNIX platforms making them unusable to users of spreadsheets from the major commercial vendors (Ballinger et al., 2003b; Igarashi et al., 1998).

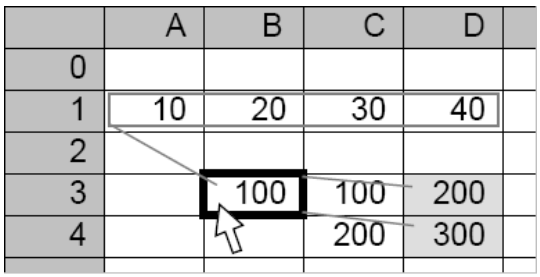


Figure 2.7 Transient Visualisation of a Cell from Ballinger et al. (2003b)

In the research conducted by Clermont et al. (2002) they found that cells had four categories of cell equivalence:

- Copy-Equivalence: Where "the formulas are absolutely identical (i.e. the cell contents has been copied from one cell into the other, either by copy and paste, or by retyping the same formula)."
- Logical-Equivalence: Where "the formulas differ only in constant values and absolute references."
- Structural-Equivalence: Where "the formulas consist of the same operators in the same order, but the operators may be applied to different arguments."
- No-Equivalence

By examining the equivalence between cells it is possible to detect inconsistencies in the spreadsheet that may help to reveal mistakes made during development or maintenance. One such example is where a formula is overwritten with a fixed value or with a different formula in an attempt to correct a problem (Clermont et al., 2002). To make the evaluation of cell equivalence easier, a *structural analysis tool* was developed that displays the spreadsheet structure in the form of a *treeview*. The user can expand and collapse the nodes on the treeview to allow them to zoom in on particular sections of the spreadsheet for closer examination. In 2003, Marcus Clermont extended his work on visualisations by producing a tool that graphically represented the relationships between cells in the form of *data dependency graphs* (DDG). By decomposing the relationships between cells into a directed acyclic graph and representing the same information as shading on the spreadsheet, cell anomalies in the spreadsheet flow can be seen (Clermont, 2003b).

As with many of the other spreadsheet research tools, this toolkit was developed for use with the *Gnumeric* spreadsheet on a *Linux* operating system and so cannot be used in the Windows environment with commercial spreadsheets such as Microsoft Excel or Lotus 1-2-3 (Clermont et al., 2002).

The visualisation of spreadsheet structure was extended even further by the work at Victoria University in Wellington, New Zealand. Building on the work of Igarashi, Clermont, Panko and Burnett, they investigated new ways of visualising spreadsheets and showing the hidden structures within them. They also to produced a visualisation

toolkit that was capable of being using across a number of platforms including Microsoft Windows. The researchers used the Java programming environment due to its platform neutrality and Binary Interchange File Format (BIFF) files saved using Microsoft Excel. The BIFF files were then analysed using a number of Java based tools and the results reproduced as diagrams (Ballinger, Biddle & Noble, 2003a; Ballinger et al., 2003b).

To gain an overview of the layout of the test spreadsheet a *real-estate utilisation* diagram was used that displayed the cell occupancy rate in each area of the spreadsheet. This diagram allows the examiner to gain an understanding of how the information on the spreadsheet has been laid out and whether the developer has purposely separated some information from the main body of the spreadsheet. The information can then be shown in a two or three-dimensional form (Figure 2.8) or via cluster diagrams that show where blocks of information are positioned. Having gained a general understanding of how the spreadsheet is arranged the examiner can then look at the cells and formulas in more detail using visualisations such as data dependency flow diagrams, dependency trees, precedence traces and summation maps (Ballinger et al., 2003a, 2003b).

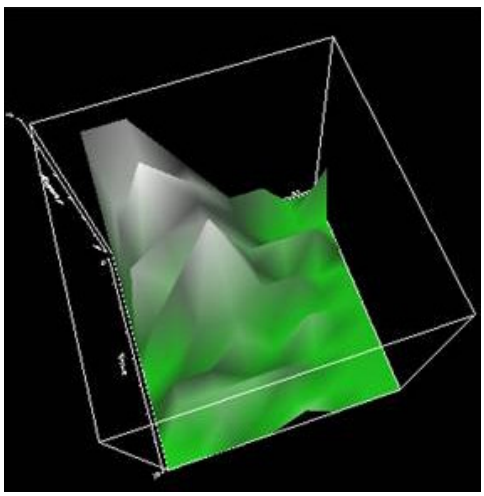


Figure 2.8 Real Estate Utilisation Diagram in 3D Ballinger et al. (2003a)

Since the aim of many of these tools is to aid in the detection of spreadsheet errors by making spreadsheets easier to comprehend (Clermont, 2003b), the broad range of visualisations and the ability to use this tool with Microsoft Excel should make this technology a useful addition to a spreadsheet reviewers toolkit.

2.3.4.5 Auditing tools

While auditing tools such as the *Spreadsheet Auditor* and *Cambridge Spreadsheet Analyst* have been available since the 1980's (Gasteiger, 1993; Knittel, 1986; Seymour, 1984), they had received very little attention from the academic community other than receiving some cautionary comments about the usefulness of the products and whether they would give a false impression of the integrity of spreadsheets.(Galletta et al., 1997; Panko, 2000a).

The lack of enthusiasm towards audit tools may be due to the belief that they cannot possibly interpret the purpose of a spreadsheet or the significance of the contents of a cell, so the tools would only ever be of limited use (Galletta et al., 1997; Pryor, 2004). As a result, it has only been in the last few years that these types of products been the focus of academic research so the quality of the tools and their usefulness to practitioners have yet to be fully evaluated (Butler, 2002).

In 2001, a paper was presented at the European Spreadsheet Risks Interest Group (EuSpRIG) on an evaluation performed on five spreadsheet audit tools: Microsoft Excel's inbuilt audit tool, Spreadsheet Professional, Excel Auditor, Spreadsheet Audit for Customs and Excise (SpACE), and the Klagenfurt Tool Kit.

The paper aimed to answer three questions about spreadsheet auditing software:

i) Is it valuable to use spreadsheet-auditing software?
ii) In what ways do the tools help users to find spreadsheet errors?
iii) What types of spreadsheet errors do the audit tools find?

To determine the answers to these three questions the evaluation applied a three-stage process to each product: i) the examiner took time to read the documentation and examine the features of the product. ii) the software was used to audit a *Sample Errors* spreadsheet and report the results and iii) the examiner reported on the results.

The evaluation did not actually test the auditing software but instead assessed how well the auditing software helped the researchers to locate the errors in the test spreadsheet (Nixon & O'Hara, 2001). This model of testing suited the *Spreadsheet Detective* which promotes itself as a tool that aids reviewers in their spreadsheet auditing role (SSD, 2003), the *Operis Analysis Kit* (OAK) which helps the user identify suspect cells and the UK Customs and Excise *SpACE* tool which requires the

user to attend a training course before using the software. The tools which did not fare as well under this model were Excel's in-built auditing functions and the *spreadsheet auditor* software (Nixon & O'Hara, 2001).

Nixon and O'Hara (2001) observed that "the most essential ability for a software tool to possess is the ability to recognise formula schema" as it means that a cell-by-cell examination of the spreadsheet should not be required. However the value of schema may need to be investigated further as schema changes may also suggest that the spreadsheet is in error when it is not and a lack of schema changes may suggest that the spreadsheet is correct whereas the formula may be the same but incorrect.

From this study the conclusion was drawn that spreadsheet-auditing software tools are useful with some of the more sophisticated tools helping the researchers to find 80%-84% of the errors in the sample spreadsheet. The reviewers also noted that the most successful audit tools used a combination of schema identification, visualisation and searching/reporting features. They also concluded that while the tools were successful in identifying qualitative, quantitative logical and quantitative mechanical errors, they did not perform very well when required to locate omission errors (Nixon & O'Hara, 2001).

The research by Nixon and O'Hara and reported usage of audit tools by the Federal Drug Administration (FDA) in the USA (Cantellops, Bonnin & Reid, 2004) and H. M. Customs and Excise in the UK (Butler, 2000) suggest that these tools may be reaching a level of sophistication that makes them a worthwhile addition for spreadsheet auditors. However, they will need much more research conducted on them before they can be declared to be "safe and effective" (Panko, 2000b).

2.4 Conclusion

The desire by end users to have their own development environment free of IT constraints was realised with the arrival of VisiCalc and other spreadsheet software. Unfortunately spreadsheets have created their own problems and since their arrival research has confirmed that most spreadsheets contain errors, the errors may be present in many different forms, are difficult to prevent and difficult to detect.

Many techniques have been used to try to deal with the problem of spreadsheet errors including development methodologies, peer review and assistance, development aids and spreadsheet auditing tools. However, none of these techniques are a complete solution to the spreadsheet error problem.

Spreadsheet audit tools have had little exposure in the past either within the commercial or academic arenas. However, with the increase in use of these tools and the beginnings of research into spreadsheet auditing software, they may prove themselves worthwhile some time in the near future.

3. RESEARCH PROCEDURE

3.1 Introduction

Research has confirmed the high incidence of errors in spreadsheets (Panko, 1998) but has failed to identify a specific solution which would significantly reduce or eliminate them. Despite spreadsheet audit tools only receiving a few cursory comments in research papers prior to 2002 (Galletta et al., 1996; Panko, 2000a; Teo & Joo Eng, 2001), Nixon and O'Hara's paper raised the possibility that audit tools may be the solution to the spreadsheet error problem (Nixon & O'Hara, 2001).

To examine this possibility further, this chapter gives the aims of the study and proposes a single hypothesis. The research methods selected to test the hypothesis are examined followed by the methodological procedures used to perform the research. The chapter finishes by examining the validity and reliability, ethical considerations and limitations of the study.

3.2 Aims

This study aims to show that participants and spreadsheet audit tools will highlight different types of spreadsheet errors. To achieve this, the participants and audit tools are subjected to the same test and the results analysed. If it can be confirmed that spreadsheet audit tools detect different errors to participants, it helps to establish the role of audit tools as a technique for reducing spreadsheet errors.

3.3 Hypothesis

For spreadsheet auditing tools to be useful to spreadsheets users, they must be able to detect errors that users tend not to find and by doing so reduce the number of errors still present in the spreadsheet. This study aims to show that spreadsheet audit tools are useful to spreadsheet users by confirming a single hypothesis:

Spreadsheet audit tools will find different errors to those found manually.

3.4 Research Method

3.4.1 Design

A contrived setting was developed to obtain two independent samples; one sample from sixty-five volunteers and the other sample from eleven spreadsheet-auditing tools. The volunteers were exposed to an *experimental simulation* in the form of an Internet web site constructed to mirror the workings of a Microsoft Excel spreadsheet and the spreadsheet auditing tools were tested against the real spreadsheet from which the web site was derived. The independent variable for the experiments was the entity (human or software) examining the spreadsheet and the dependent variable was the spreadsheet errors that were detected.

The benefit of using a contrived setting is that it allows the researcher to manipulate the independent variable and be confident that any change to the dependant variable is a consequence of that manipulation (Cooper & Schindler, 2001; Jenkins, 1985). The second benefit is that it allows the researcher to maintain tight control over the experimentation environment so that the impact of external factors can be minimised. A third benefit is that the experiment can be replicated in its original form or using different sample groups and variable settings (Cooper & Schindler, 2001; Jenkins, 1985).

The disadvantage of a contrived setting is that because it is an unnatural and controlled environment (Cooper & Schindler, 2001) some participants may find it unrealistic and intimidating and as a consequence change the way they act during the experiment (Coolican, 1996). A second related problem is that the results may be a product of the experiment environment and not reproducible outside that environment (Coolican, 1996). Thirdly, the cost of setting up a contrived environment may exceed the budgets of other forms of experimentation (Cooper & Schindler, 2001).

Given these advantages and disadvantages, a contrived environment was selected because it meant that all participants would have the same experience of the simulation, no-one benefited from using Excel's built-in auditing and formula editing features, the contents of the spreadsheet would only be available to be viewed during the period of the experiment and the time the experiment was available could be tightly controlled.

As the experiment was supposed to emulate an actual spreadsheet, it was important to keep the appearance and basic functionality of the simulated spreadsheet as close as possible to the original. To ensure that was the case, the web page was built using a hypertext-mark-up-language (HTML) export of the real spreadsheet, the formulas were exported automatically as JavaScript and the name box and formula bar were positioned in the same locations as they appear in an Excel spreadsheet. This produced a web page which closely matched the original spreadsheet in appearance and displayed the formula whenever a cell was selected as it does in Excel.

Excel features such as the menu, toolbars and function keys were not included in the simulation as they were unnecessary for the experiment and may have been a distraction for the participants. Excel's built-in auditing and formula editing features were also omitted from the simulation so that the participants would have to locate the errors themselves without the aid of the software.

3.4.2 Participants

The participants for this study were drawn from a convenience sample of people associated with the researcher. The first group of volunteers were located by sending an introductory email to 306 employees of a medium sized risk insurance company in New Zealand between the 29th of March and the 2nd of April 2004. The email explained the purpose of the study and asked the recipients if they had experience using Microsoft Excel and if so, would they be willing to assist with an experiment. Responses and email receipts were received over the next month and by late April 2004, twenty-two people had indicated that they were willing to participate in the research, thirty-five had declined, thirty-five deleted the email unread and the remainder did not respond.

The poor response rate from the first batch of emails combined with the long lag time receiving responses called into question the benefit of continuing to use emails as a recruitment technique. Therefore, on the 26th of April 2004, phone calls were made to twenty-five people known to be spreadsheet users. Some of those contacts recruited other people and because of the snowball effect a further seventy-five people agreed to participate in the research bringing the total to ninety-seven.

The spreadsheet audit tools used in this study were Excel's in-built auditing facility and those spreadsheet-auditing tools which did not require an external tester or special training before they could be used and were available as full-function demonstration software that could be downloaded via the Internet. These two criteria eliminated *SpACE*, the UK Customs and Excise auditing tool, the *Dealmaven toolkit* and the *Excel Auditor*, an audit and development tool from BYG Software.

As with the Galletta et al. (1996) study, the performance of the participants was assessed by recording the number of spreadsheet cells they identified that contained errors and the number of false positives, where valid cells were incorrectly identified as having errors.

3.4.3 Materials

The *experimental simulation* was developed using Microsoft's Visual Studio VB.NET programming language, hypertext mark-up language (HTML) and JavaScript and hosted on a Microsoft Internet Information Server (IIS). The web site could be accessed via the Internet and was available twenty-four hours a day to allow the participants to complete the exercise whenever it was convenient for them.

The web site consisted of five web pages: A sign-on page, demographics page, pre-test information page, the spreadsheet experiment page and a post-test evaluation page. The web pages (shown in Appendix B) performed the following functions:

Sign-on Page

The *sign-on* page checked whether the participant was using an Internet Explorer web browser version 5.0 or higher and the browser had JavaScript enabled. If both checks were true, an edit box was enabled that allowed the participant to enter an ID number and proceed to the next page. If either check was false the participant could

not proceed with the experiment but was given the option of contacting the researcher by email if they had any questions. The ID number was used to ensure that only those people who were authorised to access the experiment would be able to do so.

Demographics Page

The demographics page was used to collect some basic demographics about the participant including age, gender, education levels and spreadsheet experience. While the demographics were not the focus of the study, this study was modelled after the study by Galletta et al. (1996) so similar demographics were collected in order that a comparison between the studies could be conducted if necessary.

Pre-test Information Page

The pre-test information page explained how to look for and record the errors on the spreadsheet cautioned against the use of the browser refresh button and explained how to exit the test page correctly.

Spreadsheet Experiment Page

The spreadsheet experiment page was where the participants could click on cells and examine the formulae they contained and if an error was found, record the details at the bottom of the screen.

The Post-test Evaluation Page

The post-test evaluation page used Likert scales to gather the participant's estimation of the difficulty of the exercise and how well their spreadsheet experience level matched the experiment. Once the participant had answered these questions and clicked the *save and exit* button, the browser was redirected to the default home page configured on the participant's browser.

The spreadsheet audit tools were tested against a spreadsheet developed using Microsoft Excel 2002 and based on the Lotus 1-2-3 spreadsheet developed for the study by Galletta et al. (1996). The same spreadsheet was also used to generate the JavaScript code for the website formula display so that the formulae in the spreadsheet and on the website would be consistent.

3.4.4 Procedure

The procedure for the participants started when the initial contact was made with them, they received a copy of the thesis invitation sheet (Appendix D) and were asked to contact the researcher if they were happy to participate. When the participants responded, they were then emailed the follow up *scenario sheet* (Appendix E) which explained the scenario and gave the participant a unique ID number they could use to access the web site.

When the participant accessed the web site they were automatically directed to the sign-on page where they entered their ID number, and clicked the blue *GO* button. This action generated a file on the web server that recorded that the ID number had been used and the date and time the activity occurred.

After entering some basic demographic information and reading the instructions on how to perform the experiment, the participant was directed to the spreadsheet page. Upon entering the page the web file was updated to prevent the spreadsheet page being accessed a second time using the same ID number and a forty-minute timer started which displayed a "time remaining" message and the time remaining on the browser status bar. The participant was then able to click on the cells and examine the formula in them. Whenever they believed they had found an error, they doubled clicked on the cell and entered a brief description of the error at the bottom of the screen.

After ten minutes an *exit spreadsheet* button was enabled that allowed the participant to leave the experiment early if they believed that they had found all the errors in the spreadsheet. To prevent the participant from accidentally leaving the experiment, a dialog box was displayed if the exit button was clicked that asked them to confirm that they did wish to exit the spreadsheet page and if they agreed they were sent to the post-test web page.

After thirty-nine minutes had elapsed, a progress bar was added to the end of the status bar message that reduced by one segment for each of the remaining sixty seconds. Twenty seconds before the end of the experiment the browser gave a beep every two seconds until the experiment finally completed, at which point the participant was automatically transferred to the post-test web page.

On the post-test page two questions were asked to evaluate the participants' feelings about the experiment and whether they felt their Excel experience was adequate for the task. After answering the questions and clicking the *Save and Exit* button the experiment was completed and the participant's browser displayed the participant's default home page.

Three factors lead to some attrition in the number of participants completing the experiment. The first was the use of the browser's *refresh* button. When the web site was first created, using the refresh button would discard any existing data, reload the web page and reset the countdown timer back to forty minutes. As this would have allowed some participants to view the spreadsheet web page for longer than the forty-minute maximum, the spreadsheet web page was changed to exit the experiment if the refresh button was used. The second problem occurred if the participant closed the browser window instead of using the buttons provided. Like the refresh button, closing the browser window also had the effect of discarding all the data entered by the participant but also terminated the experiment. The third was intermittent network problems that some participants encountered when accessing the web site. Out of the seventy-eight participants who attempted the experiment, nine accidentally closed the browser window, one clicked refresh and three others had network problems which left sixty-five successful attempts.

Even though the test spreadsheet was modelled after the spreadsheet used in the experiment by Galletta et al. (1996), their experiment did not involve spreadsheet audit tools so the model for testing the audit tools and evaluating their performance followed the steps guidelines from the experiment by Nixon and O'Hara (2001):

i) Gain familiarity with the software.

   Each product was installed into its own directory then the help files and documentation were studied. Some time was then spent experimenting with the software to confirm that the software worked as documented.

ii) Test the software against the sample spreadsheet

   A copy of the MIS.xls test spreadsheet was copied into the same directory as the software then the functions in the software that handled spreadsheet mapping and error detection were run against the test spreadsheet.

iii) Document the results of the test and any issues with the software

The results of the test were saved to the install directory and any additional information was written to a *notes* document.

After the testing had been completed, the results were examined to ascertain how many of the errors the audit software had detected, the number of cells where errors were identified but were not present (false positives) and how accurately the software located and described the error it had found.

For the software to be credited with finding an error, it had to highlight a cell using a text description, an annotation, a cell note, a spreadsheet map or some other method that makes the error more obvious, such as applying cell formatting to reveal the underlying problem with the cell or listing a range of cells that may contain an error. This meant that simply producing a list of all the used cells in the spreadsheet, along with their contents, formatting and references was not sufficient for the audit tool to be credited with finding the error as nothing was specifically highlighted.

The reason for this style of marking was that the objective was to measure the software's contribution to the error finding process and so it was necessary for the audit tools to do something extra to indicate that an error may be present. By putting the requirement on the audit tool to identify the error, the degree of subjectivity in the test was significantly reduced as many of the tools used similar techniques and so could be evaluated on an equivalent basis.

Reliability and Validity

Whenever something is measured it is important that the measure is consistent and stable (reliability) and that it measures what it was supposed to measure (validity) (Coolican, 1996). In this section we look at the reliability and internal and external validity of this experiment.

3.4.1 Reliability

Reliability is the level to which an experiment gives consistent results and is free of the influence of random events (Cooper & Schindler, 2001). The testing methodology used in this experiment has been utilised previously by Galletta et al.

(1996) and has been shown to measure accurately the extent to which participants are able to locate spreadsheet errors. A difficulty which was presented by this test was that an Internet based experiment leaves open the possibility that the participants could be interrupted during the forty minutes the test was running. While there is no evidence that any interruptions occurred, it is a situation which could occur in real life or when the experiment was repeated so it should not affect the reliability of the experiment. A similar situation occurred in Panko's 1996 paper where he allowed his students to take the experiment home because they had little incentive to cheat on the experiment and Panko believed that it would add real world realism to the experiment.

3.4.2 Internal Validity

Internal validity exists where it can be shown that the changes to the dependent variable were caused by changes to the independent variable and not some other factor (Coolican, 1996). There are many threats to internal validity including the impact of time on participants, changes to the researcher or instrument used, participants changing their behaviour due to previous tests and contact between participants. (Cooper & Schindler, 2001). The threats to the internal validity of the experiment appeared to be instrumentation, selection and diffusion.

The instrumentation threat was present because the participants and audit software were tested using different instruments. The participants used an experimental simulation and the audit tools used a real Excel spreadsheet. However, the difference in testing instruments was used to ensure that the participants could only find the spreadsheet errors using manual checking without the help of Excel's error checking, range finder or auditing facilities.

The selection threat was present due to the different methods used to recruit volunteers but since the comparison is between participants and audit tools, not between the participants themselves, the method of recruitment should not be an issue.

The third threat to validity was the possibility that the participants could communicate the results of the test to each other. In case anyone contemplated sharing the experiment answers with others, the follow up scenario sheet (Appendix

E) stressed the importance of keeping the experiment details secret and the experiment software made saving or copying the spreadsheet difficult; however, the main factor preventing people sharing information is that all the participants were volunteers so there was nothing to be gained from cheating.

In case unintentional information sharing had occurred, the list of cells participants had visited was examined to see if any of the participants went directly to an error cell. No experiments were found in which the participants went directly to a cell containing an error however one participant did manage to locate an error after examining only three cells by checking all the SUM cells first and then moving on to the non-SUM cells. This appears to confirm that none of the participants were aware of the locations of the errors before they attempted the experiment.

3.4.3 External Validity

External validity exists where the study can be generalised to other times, participants and situations (Coolican, 1996). Threats to external validity include the effects of pre-testing; the test group being different from the general population and the differences imposed by the testing environment. In this experiment the change to the selection method may have been beneficial to the experiment by drawing on a greater cross section of the community instead of limiting the test population to people who work for a particular insurance company. The testing environment deliberately omitted the error-checking, range finder and auditing facilities available in Excel to ensure that only the participant's error detecting ability was being tested not Excel's. It could be argued that this also introduced a threat to the external validity of the experiment by making it harder for the participants to locate errors. However, the E17, H25 and G7 cells which would have been easier to detect if the range finder facility were present were also the second-equal and sixth most commonly found errors of the fifteen present in the spreadsheet (Appendix F) so it appears that omitting these features did not have a discernable impact on the experiment.

3.5 Data Collection

To conduct empirical research it is necessary to utilise one or more data collection methods in order to gather the information necessary to test a theory (Coolican, 1996; Cooper & Schindler, 2001). In this study the main methods of data collection utilised were a laboratory experiment and an experimental simulation. The laboratory experiment was used to test the spreadsheet audit tools and the experimental simulation was used to test the participants.

The experimental simulation also included a pre-test and a post-test survey which was used to capture some quantitative and qualitative data that could be compared with the error detection experiment conducted by Galletta et al. (1996).

To prevent transposition errors from occurring, the data from the experimental simulation was automatically saved to an XML (extensible mark-up language) file each time the participant moved from one web page to the next. The structure of the XML file meant that the information could be imported directly into Microsoft Excel without any need for re-keying. The results from the audit tool testing were saved as Excel spreadsheets and text files so that they could be examined later and the results copied to a results spreadsheet and eventually to Minitab.

3.5.1 Quantitative Data

Quantitative data consists of properties or qualities that can be expressed in numerical terms. In this experiment quantitative data was collected from four sources: i) the participants' responses to the pre-test questions ii) the participants' responses to the post-test questions iii) the incorrect cells identified during the experimental simulation and iv) the incorrect cells identified by the audit software.

3.5.2 Qualitative Data

Qualitative data is data that exists in textual rather than quantitative form. The qualitative data collected from the experiment were the descriptions of the spreadsheet errors provided by the participants in the experimental simulation and by the audit software from the laboratory experiment.

3.6 Ethical Considerations

The experimental simulation was conducted under the supervision and guidance of Massey University. The university also reviewed the web site and questionnaire to ensure they conformed to university requirements and the web site was developed in accordance with Massey University web site design guidelines even though the web site was hosted externally to the university.

Since the first group of participants were drawn from the staff members of a medium sized risk insurance company in New Zealand, written permission was obtained from the company's human resources department before any emails were dispatched. The human resources department also reviewed the email, web site and questionnaire.

Participation in the research was voluntary and the invitation sheet (Appendix D) and web site information page (Appendix B) detailed the purpose of the research, assured potential participants of the anonymity of their data and informed them that results would only be published in aggregate form. The participants were regularly reminded that they had the option of withdrawing from the research or having their names removed from the follow-up or informational mailing lists.

At the end of the data collection phase, the names of the participants and the names of the audit tools were removed from the data. The decision to remove the audit tool names was taken after one of the audit tool suppliers declined to participate in the experiment due to the treatment they had received during a previous research project.

3.7 Limitations

The use of an experimental simulation was one factor that must be considered as a limitation of this study. By simulating Excel, the *feel* of the test environment was slightly different to what you would find when using Excel. While none of the participants found the differences to be a problem, one did comment that he would have liked to have had access to the F2 (formula edit) button.

Another limitation of the experiment simulation occurred because of the use of web technology. Some users forgot that they were supposed to use the on-screen buttons to navigate through the experiment and not the browser's *refresh*, *back* and *windows*-

lose buttons. Using the *refresh* or *back* buttons or closing the browser window had the effect of terminating the experiment and was a significant factor contributing to the thirteen people who failed to complete the exercise. It also appears that one participant may have been prevented from completing the experiment due to web server difficulties.

Ideally the number of participants should have been higher but the group who initially agreed to participate in the experiment reduced their level of co-operation after the research stage commenced. Due to their concerns that too many people would participate in the experiment and impact on company productivity they restricted the participants to company staff instead of corporation staff. The response rate from the smaller group was so small and took so long to collect that the decision was made to discontinue asking company staff and resort to other means of finding participants.

3.8 Summary

In order to show that participants and spreadsheet audit tools tend to find different spreadsheet errors, sixty-five participants were tested using an experimental simulation and eleven spreadsheet audit tools were tested using a sample spreadsheet modelled after a 1996 experiment by Galletta et al.

The participants were tested using a simulation of a spreadsheet developed using HTML and JavaScript and hosted on a web server accessible via the Internet. The participants were given forty-minutes to examine the web page and locate fifteen errors in the spreadsheet then provide a description of each error they found.

The spreadsheet audit tools were tested against the spreadsheet used to develop the web site and according to the guidelines prescribed by Nixon and O'Hara (1996). The results were then examined to determine whether the software had detected any errors and to evaluate the quality of the detection.

The potential threats to the reliability of the experiment were the testing method and experiment interruptions. The method of testing had been used previously by Galletta et al. (1996) and was shown to be an accurate measure of participant's ability to locate spreadsheet errors and the possibility of participants being interrupted was

assessed as being minimal but if it had occurred, it would have added realism to the experiment.

The threats to internal validity were instrumentation, selection and diffusion. The instrumentation threat was due to the difference in testing instruments that were used but the difference was necessary to ensure that it was the participant and not the spreadsheet that was being testing. The different methods used to select participants would not influence the experiment and there was no evidence to support the possibility that participants may have shared experiment information with each other.

The main threat to external validity was the omission of Excel's error checking, range finder and auditing facilities from the experimental simulation however the results of the experiment appear to show that there was no discernable impact from leaving the functions out of the experiment.

A preliminary analysis of the results of the experiment appears to show that there are variations in the errors found by participants and audit tools; however this will be examined in more detail in the next section.

4. ANALYSIS

4.1 Introduction

In this section the data collected from the participants and spreadsheet audit tools is analysed and compared. The analysis presents some counts, percentages and mean values from the research, then uses a column graph, a one-sample *t* test and a two-sample *t* test to gain an overview of the data. A comparison of the individual participants and audit tool results is then performed using one-way ANOVA tests to determine whether there is any significant difference between the participant and audit tool results.

4.2 Study Analysis

4.2.1 Participants

Following an email and phone campaign, ninety-seven people agreed to participate in an experiment to compare the ability of participants and spreadsheet audit tools to locate spreadsheet errors. From the original ninety-seven, seventy-eight participants attempted the experiment of whom sixty-five completed the experiment and thirteen failed to complete it for the reasons described in the previous chapter.

The group of participants were made up of 17 (26%) females and 48 (74%) males (Figure 4.1) with the ages of the female participants ranging from 21 to 53 with a mean age of 34.3, the ages of the male participants ranging from 15 to 58 with a mean age of 38.0 and an overall mean age of 37.0 (Figure 4.2).

GENDER	Count	Percentage
F	17	26%
M	48	74%
Total	65	

Figure 4.1 Gender of participants

AGE	Up to 19	20-29	30-39	40-49	50+	Average
F	0	6	5	5	1	34.3
M	2	14	6	19	7	38.0
Total	2	20	11	24	8	37.0
%	3%	31%	17%	37%	12%	

Figure 4.2 Age of Participants by Gender

Two (3.1%) of the participants did not have any formal qualifications, Forty-five (69.2%) possessed secondary school qualifications, four (6.2%) had a tertiary qualifications and the remaining fourteen (21.5%) had overseas or other qualifications (Figure 4.3).

QUALIFICATIONS	None	Secondary School	Tertiary	Other
Participants	2	45	4	14
%	3.1%	69.2%	6.2%	21.5%

Figure 4.3 Qualifications of Participants

To assist in understanding how well participants were suited to the task of doing an experiment based on spreadsheets, participants were asked for information about their experience and use of Excel and to estimate what their expertise level would be.

When asked to record the number of years they had been using Excel and estimate their weekly usage in hours, the participants gave a mean number of years of 6.754 with a standard deviation of 2.996 and a mean weekly usage of 5.323 with a standard deviation of 6.156.

Participants rated their expertise levels on a Likert scale from 1 to 7, where 1 was a novice and 7 was an expert. The mean value of the responses was 3.97 (average) with a standard deviation of 1.262.

Calculating the correlation coefficient of the participant's estimate of their expertise level and the number of years they have been using Excel gives a Pearson correlation value of 0.481 and a p-value of 0.000. Performing the same calculation for the participant's estimate of their expertise level and the number of hours they spend using Excel gives a Pearson correlation value of 0.372 and a p-value of 0.002. In the first case the p-value of 0.000 was less that 0.001 which indicates that there is a correlation between the number of years that participants have been using Excel and their estimation of their expertise level. However in the second comparison there does not appear to be a correlation between the participant's estimation of their expertise level and the number of hours per week they use Excel. So the longer a participant had been using a spreadsheet, the higher they would rate their skill level whereas the intensity of use did not produce a similar result.

After completing the experiment participants were asked to indicate using a Likert scale how difficult they thought the experiment was (Figure 4.4) and whether they thought their experience level was sufficient for the experiment (Figure 4.5). Calculating the correlation coefficient of the estimates participants gave of their expertise level and the sufficiency of their experience gave a Pearson Correlation value of 0.498 and a *p-value* of 0.000. This indicates that there is a correlation between how participants evaluated their experience level before the experiment and their opinion of the adequacy of their experience after the experiment. So the longer the participants had been using spreadsheets, the more likely they would be to regard their experience as being sufficient for performing the experiment.

1=Easy		Experiment Difficulty				7=Hard
1	2	3	4	5	6	7
0	6	14	13	14	9	7
0.00%	9.52%	22.22%	20.63%	22.22%	14.29%	11.11%

Figure 4.4 Estimate of Experiment Difficulty

1=Insufficient		Experience Sufficiency				7=Sufficient
5	3	11	8	12	10	14
7.94%	4.76%	17.46%	12.70%	19.05%	15.87%	22.22%

Figure 4.5 Sufficiency of Experience for Experiment

A correlation coefficient was calculated of the estimates participants gave of the difficulty of the experiment and the results they achieved. This gave a Pearson Correlation value of 0.016 and a *p-value* of 0.888 which indicates that there is no correlation between the participant's estimation of the difficulty of the exercise and their eventual scores

Another check was whether a participants experience level was reflected in the scores they achieved in the experiment. Calculating the correlation coefficient of the estimates participants gave of their expertise level and the results they achieved gave a Pearson Correlation value of 0.431 and a *p-value* of 0.000 which indicates that there is a correlation between experience and scores. However, comparing participants' estimates of their level of expertise against the scores they achieved in the experiment gave a Pearson Correlation value of 0.210 and a *p-value* of 0.094 which indicates that there is no correlation between the estimate of experience and scores achieved. So the longer participants had been using spreadsheets, the more

likely it was that they would score well in the experiment but their estimation of their expertise level was not reflected in their ability to find spreadsheet errors.

Eight of the errors present in the sample spreadsheet were from the study by Galletta et al. (1997) in which twenty-three participants were asked to located errors in a spreadsheet displayed on a screen using conventional formulae. In that experiment the participants found a mean of 4.2 errors with a standard deviation of 2.0 and in this study the 65 participants found a mean of 3.523 errors with a standard deviation of 2.070. While the experiments are not directly comparable due to the software and spreadsheet dimensions being different, it is consistent with Galletta's view that an increase in the complexity of the experiment will result in a lower performance by the participants.

4.2.2 Audit Tools

While the initial intention of this study was to examine the same audit tools as Nixon and O'Hara (2001), an Internet search revealed that other audit tools were available so the decision was made to examine those tools where a full function version of the software was available via the Internet. Of the fifteen spreadsheet-auditing tools initially located only eleven could be used. Three tools, UK Customs and Excise *SPaCE* software, the *Excel Auditor* and the *Dealmaven Toolkit*, were eliminated as an Internet copy of the software was not available and *XLSior* was eliminated because it was an automatic testing tool to assist users to develop spreadsheets and not an audit tool for finding errors in existing spreadsheets.

Of the tools that were selected, the most common features were formula, cell copy and cell type maps which were present in six (54.5%) of the tools, spreadsheet summaries, formula summaries and named range lists which appeared in five (45.5%) and dependency maps in three (36.3%). Many other features were present that gave each of the tools their own distinctive strengths and weaknesses.

Some of the audit tools provided lists of commonly occurring errors and listed the cells where those errors may occur. As these lists were providing general comments and were not stating that the cell actually contained an error, they were not included in the detection or false-positives counts.

4.3 Hypothesis Testing

The objective of this research was to examine the hypothesis that *spreadsheet audit tools will find different errors to those found manually*. The first check performed on the results of the participant and audit tool results was to graph the proportion of each group that located each of the fifteen errors (Appendix F). As the graph appeared to confirm that participants and audit tools did vary in the errors they found, a one-sample *t* test and dot-plot (Figure 4.6) were created using the differences between the proportion of participants and proportion of audit tools that found each error. The analysis produced a mean of 0.1056 and a 95% confidence that any difference will be between -0.0799 and 0.2912. Since the dot-plot contains values above and below that range it shows that there are differences in the errors located by the two groups.

Performing a two-sample t test gave a mean for participants of 0.372 with a standard deviation of 0.205 and for audit tools, a mean of 0.267 with a standard deviation of 0.249. The test gave a 95% confidence that the difference in the proportion of errors found by the two groups will be in the range -0.0652 to 0.2764. The analysis also gave a p-value of 0.215 which means there is insufficient evidence to reject the hypothesis.

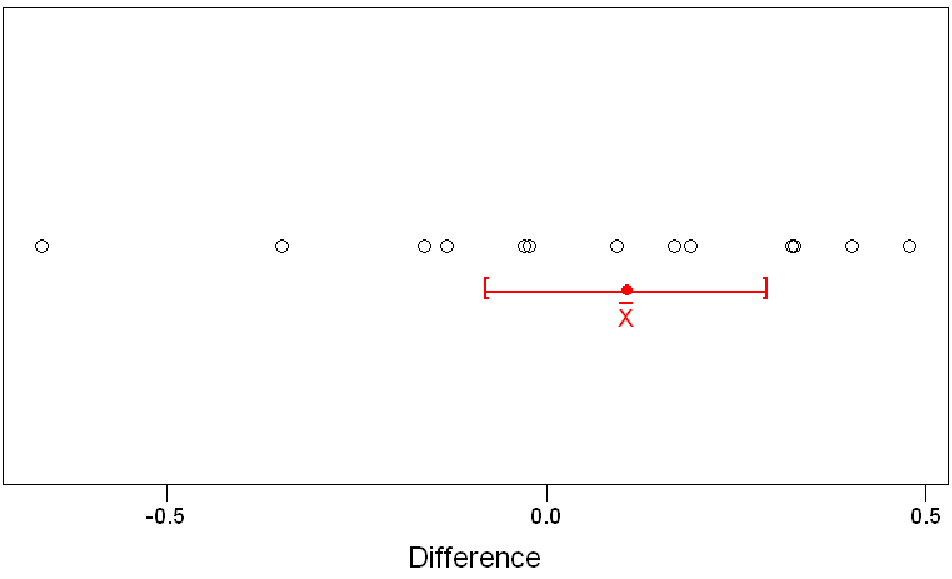


Figure 4.6 Dot-plot of Difference

To test the results further, a one-way analysis of variance (ANOVA) with the significance level set to 0.05 was used to compare the success rate of participants and audit tools finding each of the errors present in the sample spreadsheet.

The analysis of variance is a parametric test used to test the null hypothesis that the means of two or more populations are equal. The test compares the mean of a group and the values that deviate from that mean with an overall *grand mean* calculated from all the groups being examined. By comparing the variation in the means *within* the groups, against the variation *between* the groups and factoring in the number of items in each group, it can indicate whether the variations are due to chance or they indicate statistical significance (Cooper & Schindler, 2001; Hopkins, 2000).

In each of the tests which follow, the location and type of error is given, each error is described briefly, an ANOVA analysis is performed to determine if there is statistical significance in the results and any observations about the ability of the participants and audit tools to identify the error are made.

4.3.1 B31: Incorrect ISBN number

In cell B31 the value being sought by the VLOOKUP function does not exist in the range being searched and the range lookup parameter has been omitted so an approximate match of the target value will be returned (Walkenbach, 1999). It is not possible to detect the error from the value returned from the VLOOKUP as an approximate match will return the next largest value that is less than or equal to the value being sought, which in this case is the correct ISBN number.

Only four participants (6.15%) and one audit tool (9.09%) detected the B31 error. A one-way unstacked ANOVA test showed the probability of getting an *F ratio* greater than 0.13 was 0.721 (72.1%). As the *p value* of 0.721 is greater than 0.05 the data does not support rejecting the null hypothesis (H_0) that these samples are equal.

The low detection rate of this error by participants is understandable as they would need to have some understanding of how the VLOOKUP function worked and there is only a single digit difference between the ISBN lookup-value 1-56884-322-4 and the target lookup-array value 1-55884-322-4.

What is surprising is that only one audit tool singled out B31 as being a potential error and it got the error wrong by flagging B31 and the other eighteen ISBN numbers as being a *numbers formatted as text*. Since audit tools can easily identify the lookup value and check it against the lookup range, it is surprising that no audit tool correctly identified the error.

Two audit tools displayed warnings that the range being searched in a VLOOKUP function must be ascending order. Unfortunately, that statement is only true if the last parameter in the VLOOKUP function is either omitted or TRUE. If the last parameter has been specified as FALSE, the search range can be unsorted and the spreadsheet will perform an exact match and return #N/A if an exact match is not found (Walkenbach, 1999).

4.3.2 C22: Meals value entered as 11 instead of 12

Cell C22 takes the average meal cost and multiplies it by thirty days per month. The error in the formula is the use of eleven dollars instead of twelve as detailed in the *scenario sheet* (Appendix E) which stated that the average meal cost per day was twelve dollars based on two dollars for breakfast, four dollars for lunch and six dollars for dinner.

Twenty-one participants (32.31%) and no audit tools detected the C22 error. A one-way unstacked ANOVA test showed the probability of getting an *F ratio* greater than 5.11 was 0.027 (2.7%). As the *p value* of 0.027 is less than 0.05 the data does support rejecting the null hypothesis (H_0) that these samples are equal and the result can be viewed as being significant (Coolican, 1996).

As the detection of this error depended on the searcher using the average meal cost information supplied in the *scenario sheet* (Appendix E), it is logical that the audit tools would fail to detect the error as they could not utilise the information.

4.3.3 E17: Incomplete range used

Cell E17 was supposed to select the total of all the books to be purchased and then divide the value by three to give the amount to be spent each semester. Instead the SUM formula selected four of the five books, omitting the value for the *MS Excel Function Reference* book in cell C32.

Thirty-seven participants (56.92%) and one audit tool (9.09%) detected the E17 error. A one-way unstacked ANOVA test showed the probability of getting an *F ratio* greater than 9.45 was 0.003 (0.3%). As the *p value* of 0 003 is less than 0.05 the data does support rejecting the null hypothesis (H_0) that these samples are equal and the result can be viewed as being highly significant (Coolican, 1996).

Thirty-five participants identified the error but two of them then suggested corrections to the formula which would have resulted in the formula still being wrong. One of the corrections would have been rejected by Excel as it was syntactically incorrect but the other would have been accepted and produced an incorrect result.

The only audit tool to identify the cell as possibly containing an error, selected it because it stated that it had a high risk of containing an error as it contained a mix of formulas and values. So it did not actually identify the error but selected an example of poor coding practice instead.

4.3.4 E25: Multiplication by 25 should use cell C25

Cell E25 was supposed to multiply the twenty-one dollar monthly allocation for clothing by the four months in each semester. Instead the "C" was omitted from the front of the C25 cell reference and instead of being multiplying by the twenty-one dollars in C25, it multiplied the four months by the number twenty-five producing a result sixteen dollars higher than expected.

Thirty-six participants (55.38%) and four audit tools (36.36%) detected the E25 error. A one-way unstacked ANOVA test showed the probability of getting an *F ratio* greater than 1.35 was 0.248 (24.8%). As the *p value* of 0.248 is greater than 0.05 the data does not support rejecting the null hypothesis (H_0) that these samples are equal (Coolican, 1996).

All the participants who identified the cell as being incorrect and then suggested an alternative formula provided a formula that would have produced the correct result. Four audit tools also highlighted the cells as possibly containing errors but two of them picked the cell because the cell formula was different from surrounding cells and two others indicated that the C25 cell had not been used.

4.3.5 F10: Subtraction of Cell F4 is omitted

Cell F10 does not include the code to subtract the cash available at the beginning of the semester in cell F4.

Thirty-eight participants (58.46%) and two audit tools (18.18%) detected the F10 error. A one-way unstacked ANOVA test showed the probability of getting an *F ratio* greater than 6.48 was 0.013 (1.3%). As the *p value* of 0.013 is less than 0.05 the data does support rejecting the null hypothesis (H_0) that these samples are equal and the result can be viewed as being significant (Coolican, 1996).

Two of the participants who identified the cell as being incorrect and then suggested an alternative formula provided a formula that was incorrect. As both alternatively were syntactically correct, Excel would have accepted them and produced an incorrect result. Two audit tools also highlighted the cells as possibly containing errors based on the cell formula being different from surrounding cells.

4.3.6 F24: Formula uses addition instead of multiplication

Cell F24 should take the transportation value from the first semester and multiply it by one plus the rate of inflation; instead it adds one plus the inflation rate.

Twenty-five participants (38.46%) and six audit tools (54.55%) detected the F24 error. A one-way unstacked ANOVA test showed the probability of getting an *F ratio* greater than 0.99 was 0.322 (32.2%). As the *p value* of 0.322 is greater than 0.05 the data does not support rejecting the null hypothesis (H_0) that these samples are equal (Coolican, 1996).

One participant who identified the cell as being incorrect suggested that the correct formula should be "=E24*D24". This may be because cell E24 contains the formula "=D24*C24" and they assumed that the formula had been copied from the left instead of the right. Three of the audit tools identified the cells because they were different from surrounding cells and three others were located using spreadsheet shading and maps.

4.3.7 G7: Incomplete range used

Cell G7 was supposed to select the values that made up the living costs for the summer semester but it omitting the clothing value in cell G25.

Thirty-four participants (52.31%) and six audit tools (54.55%) detected the G7 error. A one-way unstacked ANOVA test showed the probability of getting an *F ratio* greater than 0.02 was 0.892 (89.2%). As the *p value* of 0 892 is greater than 0.05 the data does not support rejecting the null hypothesis (H_0) that these samples are equal (Coolican, 1996).

One audit tool highlighted the cell as possibly containing an error based on the cell formula being different from surrounding cells and fives other audit tools produced maps that highlighted that the cells as being inconsistent.

4.3.8 G11: Formula adds the wrong cell

In cell G11 the formula adds the living cost value in cell "F7" instead of "G7".

Twenty-seven participants (41.54%) and six audit tools (54.55%) detected the G11 error. A one-way unstacked ANOVA test showed the probability of getting an *F ratio* greater than 0.64 was 0.428 (42.8%). As the *p value* of 0.428 is more than 0.05 the data does not support rejecting the null hypothesis (H_0) that these samples are equal (Coolican, 1996).

As with the previous error, one audit tool highlighted the cell as possibly containing an error based on the cell formula being different from surrounding cells and five other audit tools produced maps that highlighted the cells as being inconsistent.

4.3.9 H11: An incorrect formula is used

Cell H11 sums the values from cell E11 to G11. If this formula was correct it would be useless as it would total the money remaining at the end of the three semesters. Instead it should contain the formula "=G11", the cash remaining at the end of the summer semester.

Eleven participants (16.92%) and no audit tools (0.00%) detected the H11 error. A one-way unstacked ANOVA test showed the probability of getting an *F ratio* greater than 2.18 was 0.144 (14.4%). As the *p value* of 0.144 is more than 0.05 the

data does not support rejecting the null hypothesis (H_0) that these samples are equal (Coolican, 1996).

This error was confusing for some of the participants and drew comments such as "this figure is meaningless", "this cell serves no real purpose" and "I don't think this is logical". However those same people did not then suggest what would have been logical and by doing so correct the formula.

The error was also difficult for audit tools to detect as the purpose of the spreadsheet has to be understood to conclude that the formula is wrong. Instead the audit tool found that the formula in cell H11 appeared to be a copy of the formula in H10 so it concluded that the cell was correct.

4.3.10 H25: Too many cells included in SUM range

The formula "=SUM(D25:G25)" in cell H25 is supposed to sum the clothing figures from the three semesters but it incorrectly includes the number 4 in the cell D25.

Thirty-seven participants (56.92%) and one audit tool (9.09%) detected the H25 error. A one-way unstacked ANOVA test showed the probability of getting an *F ratio* greater than 9.45 was 0.003 (0.3%). As the *p value* of 0.003 is less than 0.05 the data does support rejecting the null hypothesis (H_0) that these samples are equal and the result can be viewed as being highly significant (Coolican, 1996).

One participant came close to resolving the problem but suggested that the replacement formula should be "=E24:G25" which results in a cell error as it returns an array of six cells containing the values 160, 640, 653, 100, 102 and 104 instead of a single value and the only audit tool to highlight the cell as possibly containing an error did so based on the cell formula being different from surrounding cells.

4.3.11 H33: Operator precedence error

Cell H33 is supposed to sum the values in the entertainment options range H28 to H32 and then divide the result by the number of items to get an average. However due to operator precedence Excel evaluates the "H32/COUNTA(H28:H32)" portion of the formula before it deals with the additions which produces a result of 62 instead of 44.4.

The problem occurs because Excel interprets operators in the order of negation, percentage, exponentiation, multiplication and division, addition and subtraction, text concatenation and then comparison. To change the way Excel interprets formulae, parenthesis can be used which will make Excel resolve the innermost parenthesis first and then work its way outwards (Walkenbach, 1999).

Thirty-seven participants (56.92%) and one audit tool (9.09%) detected the H33 error. A one-way unstacked ANOVA test showed the probability of getting an *F ratio* greater than 9.45 was 0.003 (0.03%). As the *p value* of 0.003 is less than 0.05 the data does support rejecting the null hypothesis (H_0) that these samples are equal and the result can be viewed as being highly significant (Coolican, 1996).

Two participants commented on the use of COUNTA, suggesting that it was the wrong function to use. However whether COUNT or COUNTA is used will make no difference as all the cells are numeric and none are empty, so both COUNT and COUNTA will produce the same result.

The low detection rate of this error by the audit tools is surprising because it is a trivial programming exercise to determine if a formula may be impacted by operator precedence. Another problem is that only a single audit tool identified the cell as being incorrect and it selected it because it claimed the formula omitted adjacent cells. Since the COUNTA function only counts numeric cells and all the adjacent cells are either empty or text, there is no reason for the audit tool to flag this is an error.

4.3.12 K9: Value is stored as text instead of a number

Cell K9 contains the value $90 but it has been entered as text. This occurs when the value is entered with an apostrophe preceding it, when the cell has been formatted as text or the value has been copied or imported from somewhere else and stored in the spreadsheet as text (Walkenbach, 1999). In the sample spreadsheet it is possible to tell that the $90 is text because it has an apostrophe at the beginning and the value is ignored by the SUM function in cell K20 and in a cell with default alignment settings a text value will left-align whereas a numeric value will right-align.

Ten participants (15.38%) and nine audit tools (81.82%) detected the K9 error. A one-way unstacked ANOVA test showed the probability of getting an *F ratio*

greater than 30.43 was 0.000 (0.0%). As the *p value* of 0.000 is less than 0.05 the data does support rejecting the null hypothesis (H_0) that these samples are equal and the result can be viewed as being highly significant (Coolican, 1996).

Determining that a cell contains a number stored as text is very easy for software, so the surprise for this cell was that two of the tools failed to detect the error. Excel's internal auditing facilities clearly identified cell K9 as containing a *number stored as text* but three other tools did not do as well. The first tool did produce a map in which the $90 figure was left aligned but since the tool did not detect or highlight the error it was not given credit for finding it. A second tool stated that the spreadsheet did not contain any cells with numbers stored as text and the third identified a range of numbers of which at least one was text then it contradicted itself by producing a map in which the same range was displayed as containing only numbers.

4.3.13 K30: COUNTIF searches for the wrong value

Cell K30 uses a COUNTIF function to count the number of *Excel* books and a second COUNTIF to count the number of *MS Office* books. The objective is to count the number of Microsoft publications that are available. Unfortunately, the second function does not find any items because the "MS Office" group name is missing the middle space character.

Thirty-three participants (50.77%) and two audit tools (18.18%) detected the K30 error. A one-way unstacked ANOVA test showed the probability of getting an *F ratio* greater than 4.13 was 0.046 (4.6%). As the *p value* of 0.046 is less than 0.05 the data does support rejecting the null hypothesis (H_0) that these samples are equal and the result can be viewed as being significant (Coolican, 1996).

The participants who identified K30 as an error fell into two groups. The majority realised that the MS Office group name was missing a space but three participants expected it to behave like the cells above and below and only search for one item.

The audit tools did not perform well with this cell as only two tools identified the cell as being a potential error then both of them listed the wrong error. The first tool indicated that cell J30 had not been used as input by K30 so would have drawn attention to K30. The second tool warned about the presence of nested IF statements

in the cell which is odd since the cell only contains COUNTIF statements and they are not nested.

4.3.14 K32: COUNTIF uses the wrong reference cell to search

Cell K32 is supposed to use a COUNTIF function to count the number of books in the "GUI" group but instead of using the "GUI" label in cell J32 it uses the "Analysis" label in cell J31.

Twelve participants (18.46%) and one audit tool (9.09%) detected the K32 error. A one-way unstacked ANOVA test showed the probability of getting an *F ratio* greater than 0.57 was 0.452 (45.2%). As the *p value* of 0.452 is more than 0.05 the data does not support rejecting the null hypothesis (H_0) that these samples are equal (Coolican, 1996).

Most of the participants who identified this error were able to provide the correct formula to fix the problem. However only one audit tool identified the cell as being a potential error due to the cell not been used as input by K32.

4.3.15 L2: Excel interprets cell value as 1904

Cell L2 uses the DATE function to create a date but due to the developer omitting the century from the function Excel has interpreted the date as 1904 instead of 2004. This error appears to be due to an inconsistency in Excel as a date been entered using "15/03/04" would be interpreted as Monday the 15th of March 2004 but entering the same date using "= DATE(04,03,15)" is interpreted as Tuesday the 15th of March 1904.

One participant (1.54%) and four audit tools (36.36%) detected the L2 error. A one-way unstacked ANOVA test showed the probability of getting an *F ratio* greater than 23.92 was 0.000 (0.0%). As the *p value* of 0.000 is less than 0.05 the data does support rejecting the null hypothesis (H_0) that these samples are equal and the result can be viewed as being highly significant (Coolican, 1996).

The only participant who identified cell L2 as being a potential error thought the DATE function had been used incorrectly as it had not been entered using American date format. No audit tool clearly identified the date in cell L2 as being a Y2K error even though one of them specifically listed Y2K checking as a feature of their

software however four audit tools were given credit for highlighting the error as they produced reports which clearly showed the date in full century format (i.e. 1904).

4.3.16 Conclusion

An analysis of the results showed that for nine of the fifteen errors, a higher proportion of participants found the error than the proportion of audit tools (Appendix F). Three of the errors had a significant difference in detection rates in favour of the participants, three had a highly significant difference in favour of the participants and two were highly significant in favour of the audit tools. This shows that while audit tools are useful, their contribution does not appear to be as significant as would be achieved using peer review.

The cells with the lowest overall detection rate of 5 (6.58%) were cells B31 and L2. These two cells required the participants and audit tools to have a good understanding of Excel's VLOOKUP and DATE functions for them to be able to ascertain that an error existed.

The cells with the highest overall detection rate of 40 (52.63%) were E25, F10 and G7. As these cells are different to neighbouring cells it should have been easy for the audit tools to detect the differences using pattern matching (O'Beirne, 2003). However a higher proportion of participants detected cells E25 and F10 (55.38% and 58.46%) than audit tools (36.36% and 18.18%) and a slightly higher proportion of audit tools detected cell G7 (52.31%) than participants did (54.55%).

Participants found the error in cell F10 (58.46%) the easiest to detect and the highest detection rate for audit tools occurred with K9 (81.82%), the $90 figure stored as text. Nine of the eleven audit tools detected this error in some way but the surprise was that two of the tools failed to detect the problem.

For both the lowest and the highest detected cells and the *number stored as text* error, it should have been possible for the audit tools to identify the errors and give a good description of the problem. Compared with participants, the performance by the audit tools was disappointing and may suggest that audit tools need a lot more work before they can be declared "both safe and effective" (Panko, 2000b).

From the results, it appears that participants find it easier to detect an error if it involves a range that is either too long or too short (20.0%) or where there were subtle changes in the formula compared to adjoining cells (26.7%) and audit tools performed best when the problem was technically wrong, such as a Y2K error, a number stored as text (13.3%) or where there were changes in the formula compared to adjoining cells (20.0%).

4.4 Summary

The data collected from the sixty-five participants and eleven audit tools were analysed using one-way ANOVA tests to determine whether there is any significant difference between them. The results show that for eight of the fifteen errors there were significant or highly significant differences between the results of the participants and those of the audit tools. The results also showed that in most cases participants outperformed audit tools even in those areas where audit tools could be expected to perform well and that the longer a participant had been using Excel, the more likely they were to find the errors. While these results may suggest that audit tools still need to be improved, it does support the hypothesis that *spreadsheet audit tools will find different errors to those found manually*.

5. DISCUSSION AND CONCLUSIONS

5.1 Introduction

The study described in this thesis built on the work of Galletta et al. (1997) and Nixon and O'Hara (2001) and aimed to show that there would be differences in the types of errors found by participants and auditing tools.

This section briefly summarises the methods used to test the hypothesis and the results that were obtained. The implications, limitations and value of the study are then examined and some observations about the experiment are made and suggestions for further research in the area of spreadsheet errors are given.

5.2 The Study's Hypothesis

The hypothesis for this study was *spreadsheet audit tools will find different errors to those found manually*. To determine if this may be the case, a web site was developed so that participants could search for errors in a simulated environment and eleven audit tools were downloaded and tested against a sample spreadsheet. Examining the results with ANOVA revealed that for eight of the fifteen errors in the spreadsheet, there were significant or highly significant differences between the ability of the participants and the audit tools to find errors. These results show that when given the task of finding errors in a spreadsheet, participants and audit tools will tend to find different types of errors.

5.3 Value of the Research

Audit tools have been available since the 1980's but they have had limited success detecting spreadsheet errors and there is little prospect that a tool capable of finding all spreadsheet errors will be produced in the near future (Galletta et al., 1993; Galletta et al., 1997). There are also concerns that spreadsheet users may be lulled into a false sense of security from using spreadsheet audit tools as they may believe that a spreadsheet that has been audited are free of errors (Panko, 2000a, 2000b).

The spreadsheet audit tools tested in this study completely missed two of the errors; five errors were only detected by one tool and two errors were detected by two tools.

The best performing audit tool highlighted nine errors but it did not give an accurate description of any of the errors. By contrast, each of the errors were detected by at least one participant, the best performing participant found eleven errors and for ten of those, they gave a detailed description of the problem.

From this study it is clear that audit tools are not going to eliminate spreadsheet errors but the study did find significant differences between the errors found by the participants and those found by spreadsheet auditing tools. So despite audit tools being incapable of locating all the errors in a spreadsheet, the errors they do find are likely to be different to those found by spreadsheet users. This factor alone means that audit tools do have a role in detecting spreadsheet errors and adds credence to Nixon and O'Hara's (2001) view that spreadsheet-auditing programs are useful tools to have even if they do have their limitations.

5.4 Limitations of the Research

5.4.1 Experimental Simulation

As stated in chapter 3, a limitation of this study was the use of an experimental simulation, in the form of an Internet web site, as the medium for the participants to take part in the experiment. The experimental simulation was chosen because it allowed the experiment to be tightly controlled so that the impact of external factors could be minimised and it made it possible for this study to model closely the study performed by Galletta et al. (1997).

The use of an experimental simulation meant that Excel's error checking, range finder and auditing facilities were not available to the participants even though they would have been available under normal circumstances. One participant specifically commented that if they had been given an equivalent of Excel's F2 function (formula edit), it would have made it easier to complete the experiment. However it was considered necessary to omit these features so that the study was only measuring the performance of the participants and audit tools and not the additional features of Excel.

5.4.2 Web Problems

When the study began it was clear that it would be difficult to assemble a group of people together to perform an experiment as Galletta et al (1997) had done, so a web based experiment was selected instead. Unfortunately a web site presents its own problems as it adds a very large technology burden to the list of tasks that must be performed to complete an experiment successfully.

Due to a web site being used, quite a few problems crept into the experiment. The web server upon which the experiment was hosted was subject to intermittent network problems and may have been responsible for three participants failing to complete the experiment.

Using a web site also meant that a lot of work was involved developing the web pages, writing the web server logic, writing the client-side (browser) JavaScript, ensuring that the data could be collected and stored correctly and handling the security of the web site. With all those tasks, it is inevitable that some things would be overlooked and that was the case with this study as well. Two of the problems encountered in this study were:

iv) Incorrect questions.

On the *pre-test information page*, question three asks "what is your highest secondary school qualification?" but the options include tertiary, overseas and other qualifications. This does not appear to have had any impact on the experiment and none of the participants commented on it being a problem which is why it was only discovered once the experiment had ended.

v) Configuration problems

The web page was developed using a screen with a 760 x 1024 resolution. However, if the participant was using a different screen resolution they saw a slightly different view of the experiment. One of the first people to attempt the experiment commented that because they used an 800 x 600-screen resolution, they could not see the "Exit Spreadsheet" button on the *spreadsheet page* and so they had to wait for the experiment to time out for the screen to close. Fortunately this problem was quickly corrected and the button was moved so that it could be seen on a screen using a lower screen resolution.

From this experience it is clear that performing experiments using web technology is a far more complicated task than it first appears and it can introduce complex problems that are time consuming to resolve, that would not have occurred if a paper experiment had been conducted instead.

5.4.3 Marking Scheme

In the experiment by Galletta et al. (1997), two independent judges were used to mark the results and a third compared the marks for consistency and acting as a tie-breaker when necessary and in the experiment by Nixon and O'Hara (2001) their paper indicates that the most of the marking was conducted by David Nixon, the author.

The intention for this study was to use the graduated marking scheme used by Nixon et al. (2001) and for the results to be marked by two markers. However the marking scheme was not appropriate for both participants and audit tools so it had to be adjusted and the time constraints imposed by collecting the information from both participants and audit tools meant that it was not feasible to involve a second party in the marking process.

Nixon and O'Hara also recommended that the marking for the audit tools be conducted by participants who did not know the location of the errors beforehand however that would introduce the skill level of the markers as another variable in the marking process and would have required eleven markers so the decision was made to stay with a single marker (Nixon & O'Hara, 2001).

Ultimately, the graduated marking scheme was used but the analysis was based upon whether or not the participants and audit tools were able to point out the location of an error, regardless of the reason for the indication. This made most of the marking very simple but some subjectivity was still involved in determining how to mark the spreadsheet maps and spreadsheet summaries.

5.5 Suggestions for Further Research

As minimal research has been conducted in the area of using spreadsheet audit tools to locate errors in spreadsheets (Galletta et al., 1993; Nixon & O'Hara, 2001), there are many areas of audit software use and their benefits which have not been explored. Four areas that could be studied are: i) performing this study in a laboratory environment, ii) examining the impact of Excel's audit facilities, iii) looking at the knowledge level of participants and iv) examining the impact of cell similarity on error detection rates.

5.5.1 Laboratory environment

While the use of a web site simplified the management and control of the experiment, it also made the process more complicated by requiring the use of a lot of fragile technology. A better option would be to run this experiment in a laboratory environment such as a Massey University with the help of student volunteers. This would have the added advantage that a standard version of Excel would be available within the university, the access to the experiment could be tightly controlled by the network administrators and the results could be automatically transmitted directly from the spreadsheet to a central source for analysis.

5.5.2 Excel's audit facilities

The experiment simulation created an artificial environment where the participants did not have access to Excel's error checking, range finder and auditing facilities. While it was important for this experiment to eliminate those facilities to ensure that only the participant's error finding abilities were being measured, a university environment would make it possible to measure the impact of those facilities to ascertain whether they are helping to reduce the incidence of spreadsheet errors.

5.5.3 Knowledge levels

Comments from participants about the use of COUNTA and COUNTIF functions in the study highlighted some deficiencies in the participant's Excel knowledge. As past research has shown that many users create spreadsheets with little or no formal training (Butler, 2002; Whittle & Janvrin, 2002) and that the efficacy of people

finding spreadsheet errors was influenced by their domain knowledge and past error finding experience (Galletta et al., 1993; Galletta et al., 1996) it is possible that some spreadsheet errors may simply be the consequence of insufficient knowledge. To evaluate if that is the case, an experiment could be performed to evaluate if there is a correlation between the knowledge a person has of spreadsheet functions and those functions containing errors in the spreadsheets they produce.

5.5.4 Cell similarity

In this experiment the spreadsheet audit tools found 20% of the errors because the cells they identified were different to nearby cells. This may prove to be a problem if the spreadsheet being tested contains a high number of cells that are slightly different but still correct or where the cells are mostly different, as the audit tools may not perform as expected. A worthwhile test to perform on the audit tools would be to evaluate the impact of similar or dissimilar cells on the ability of audit tools to detect errors.

5.6 Research Observations

5.6.1 The role of audit tools

Some of the spreadsheet audit tools promote themselves as being products that assist the user to locate errors in spreadsheets and to allow them to examine the spreadsheet's structure and logic flow. This means that the user, not the software that has the primary role of finding the errors. One vendor makes the statement that it is better to have a product that assists the user to understand the hidden structure of the spreadsheet than to have one that incorrectly highlights cells as being in error and by doing so lead the user astray (EXChecker, 2003; OAK, 2001; SSD, 2003).

If the vendor promotes their audit tool as being an aid to error detection then presumably it must be doing something to aid the error detection process and it is that extra something that this study was trying to measure. The audit tools seem to accept this position by using comments like "this feature (i.e. Cell Input/Output Highlighting) is an excellent way of checking documents for errors" (EXChecker, 2003), this tool helps "to locate errors in the logic of the worksheet"(OAK, 2001) or "Inconsistent shading makes the errors in these two cells obvious" (SSD, 2003).

5.6.2 Error detection quality

While the audit tools did manage to indicate the location of errors forty-four times, only seven (15.9%) of those accurately identified and described an error, eleven (25.0%) hinted strongly at an error and twenty-six (59.1%) indicated that a cell may have an error but did not provide a description of the error or described the wrong error.

Since all of the accurate (*passed*) error identifications were for cell K9, this means that thirty-seven (84.1%) of the error detections were rated as *almost passed* or *almost failed* (Appendix A). Nine of the *almost passed* errors were "inconsistent cell" errors; one was a "formula omits adjacent cells" error and one was a "non-numeric cell in range" error for cell K9.

The *almost failed* errors consisted of sixteen (61.5%) error locations suggested by maps or formula reports, one (3.8%) incorrectly identified as being a number formatted as text, four (15.4%) were Y2K errors, one (3.8%) cautioned that E17 containing mixed formulas and values, one (3.8%) cell was incorrectly identified as containing nested IF statements and three cells (11.5%) were identified without explanation. So of the 165 error locations that could have been identified by the eleven audit tools, only 44 (26.7%) were identified and two were because the audit tool incorrectly evaluated the contents of the cell, further reduced the identification rate to 42 (25.5%).

5.6.3 Error detection methods

Some of the techniques used by the audit tools to assist in the auditing process may actually make the task more complicated:

i) Mapping symbols

One tool produced a map where an asterisk (*) was used to indicate a distinct formula but a less-that symbol (<), a circumflex-accent (^) and a plus sign (+) were used to indicate that the cell formula was the same as the cell to the left, the cell above or both. This had the odd effect of displaying three asterisks to show that the cells were different and three different symbols (less-that, circumflex-accent and plus sign) to show that the cells were the same.

ii) Confusing maps.

A mapping technique used by an audit tool highlighted every cell which contained a formula that was different to the cell to its left. However, in a spreadsheet in which most of the cells have been copied from the cells above, it produces a map that shows that many of the cells are distinct when that is not the case (Figure 5.1).

19	$1,800	$1,800	$1,800	$5,400
20	$212	$212	$212	$636
21				
22	$1,320	$1,346	$1,373	$4,039
23	$600	$612	$624	$1,836
24	$160	$161	$164	$485
25	$100	$102	$104	$310

Figure 5.1 Spreadsheet Map Problem

A second tool struggled with the use of merged cells in the spreadsheet and produced a map that showed that some cells contained formulas even though they were empty (Figure 5.2).

27		T					T		T	
28		T	F	F	F	F	T	N	T	F
29		T	^	^	F	F	T	N	T	F
30		T	^	^	F	F	T	N	T	F
31		T	^	^	F	F	T	N	T	F
32		T	^	^	F	F	T	N	T	F
33		T	F				T	F	T	F

Figure 5.2 Spreadsheet Map Error

iii) Potential error spots

Some of the tools provided a list of common trouble spots where errors were likely to occur. One tool provided a list of eleven potential trouble areas consisting of 102 cells that may be incorrect but only two of the cells listed contained an error and neither of them was for the reason given.

5.7 Conclusion

As very few studies had been conducted on the impact of spreadsheet auditing software, this thesis examined the hypothesis that *spreadsheet audit tools will find different errors to those found manually.* To evaluate the hypothesis, two experiments were conducted. The first involved sixty-five participants completing an experiment via the Internet in which they tried to find fifteen errors in a simulated spreadsheet and the second involved downloading eleven spreadsheet audit tools from the Internet and testing them against a sample spreadsheet.

Unfortunately, thirteen of the participants failed to complete the experiment due to technical difficulties. Ten accidentally terminated the experiment by clicking refresh or by closing the browser window and three others could not complete the experiment due to network problems. Despite the difficulties, the use of a web site was beneficial as the experiment was accessible through the Internet and the results were automatically saved as XML files, eliminating the possibility of transposition errors.

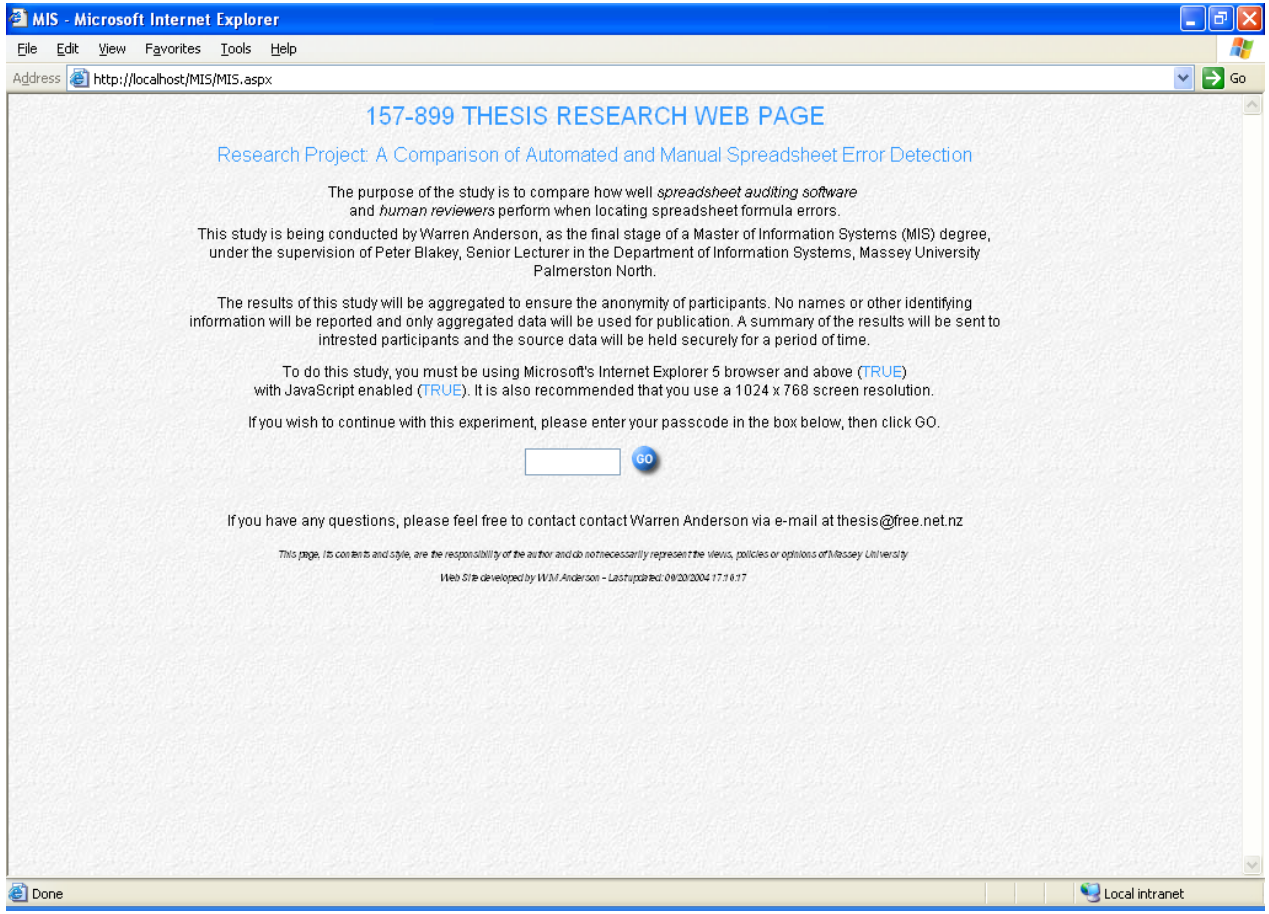
After the data from participants and audit tools had been collected, it was analysed with ANOVA which confirmed that for eight of the fifteen errors in the spreadsheet, there were significant or highly significant differences between the errors found by the participants and those found by the audit tools. The analysis also revealed that in this particular experiment the participants performed better than the audit tools, with five of the seven errors showing significance being those found primarily by the participants.

Despite the poor performance of the audit tools, they did succeed to detecting errors that the participants found difficult to find. This is significant as means that it is not necessary for audit tools to reach the level of being “both safe and effective” (Panko, 2000b) for them to be useful. Instead, this research and the work by Nixon and O’Hara (2001) has shown that spreadsheet audit tools are a useful tools to have and while they are not infallible will help to detect spreadsheet errors.

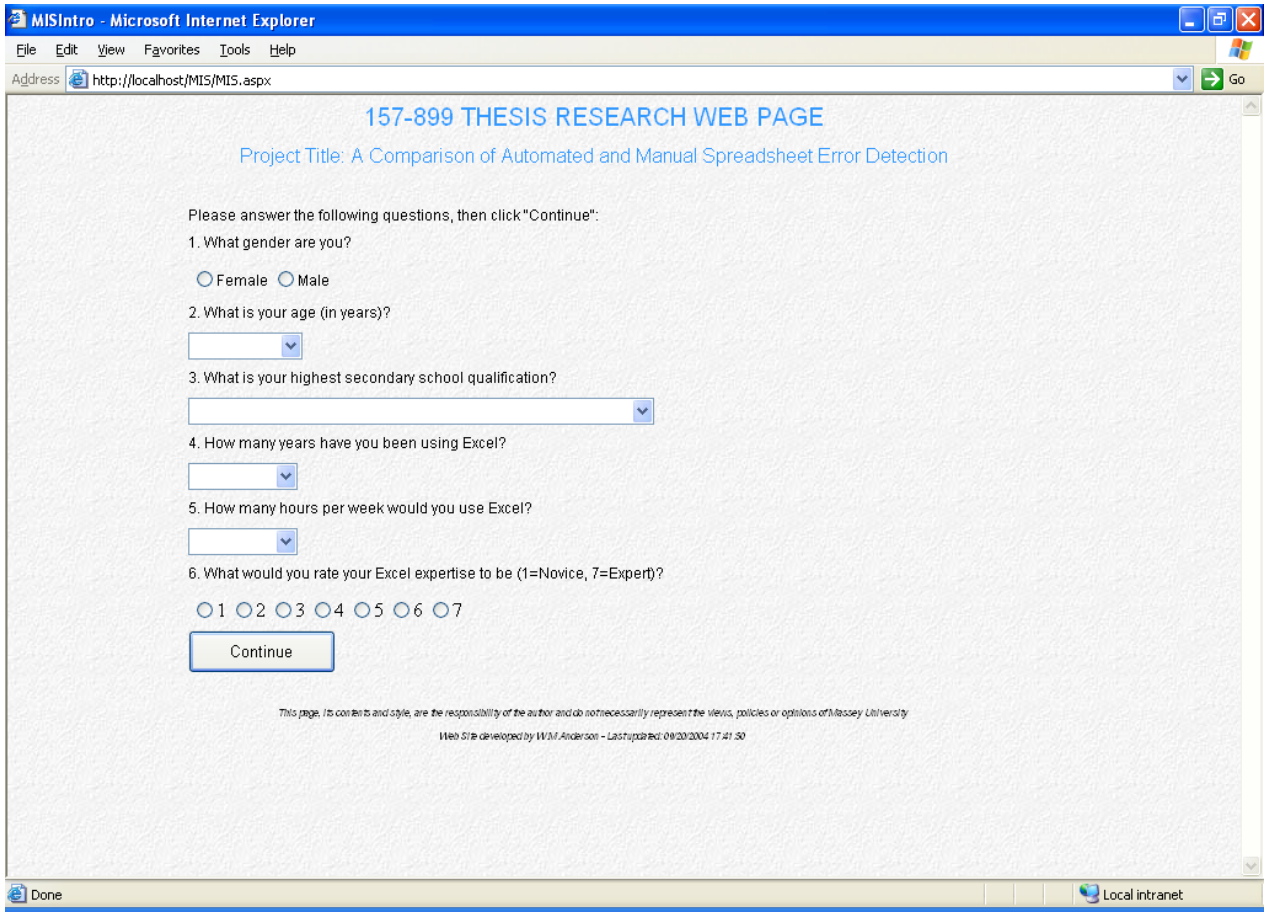
APPENDIX A – MARKING SCHEME

Rating	Rating - Nixon & O'Hara	Rating – Audit Tools	Rating - Participants
PASSED	The software spotted the error with minimal intervention from the user, often highlighting the problem before the user examined the erroneous cell(s) on an individual basis. The software made the error much easier to spot by using visual display methods.	The software identified the error, either through a description or visual display methods, and provided an accurate description of the problem	The volunteer identified the error and provided an accurate description of the problem
ALMOST PASSED	The software hinted strongly at the possibility of the error and after a little investigation, the user, aided by the display tools in the software, was able to find the error.	The software hinted strongly at the possibility of the error and after a little investigation, the user, aided by the description or the display tools in the software, was able to find the error.	The volunteer identified the cell as containing an error, but did not accurately describe what the error was.
ALMOST FAILED	The software suggested the presence of an error but failed to identify the rogue cell(s). An investigation by the user found the offending error.	The software suggested the presence of an error, but did not provide a description of the error or described the wrong error	The volunteer identified the cell as containing an error, but did not provide a description of the error or described the wrong error
FAILED	The software failed to provide any suggestion that the error existed.	The software failed to provide any suggestion that the error existed.	The volunteer did not identify that an error existed

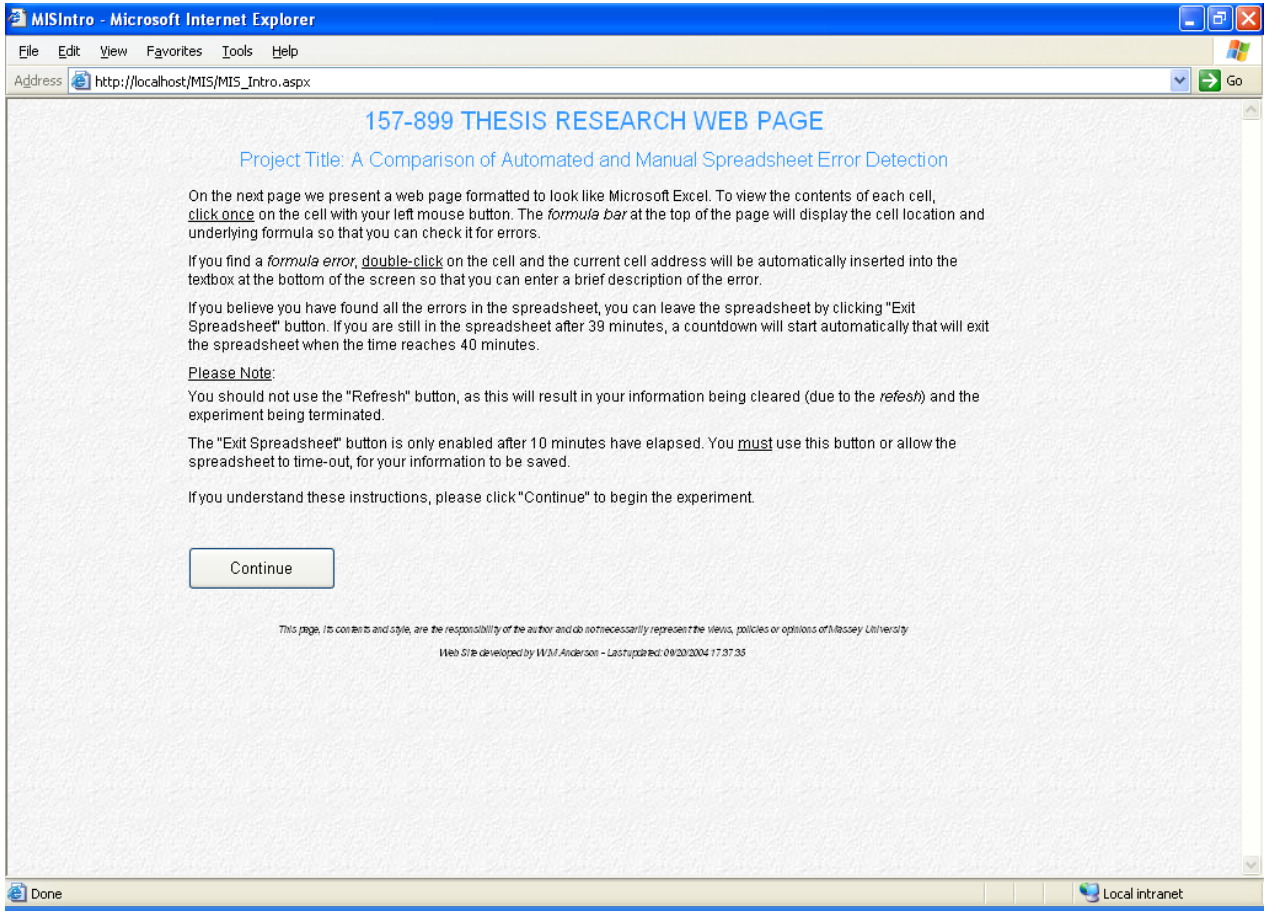
APPENDIX B – WEB PAGES



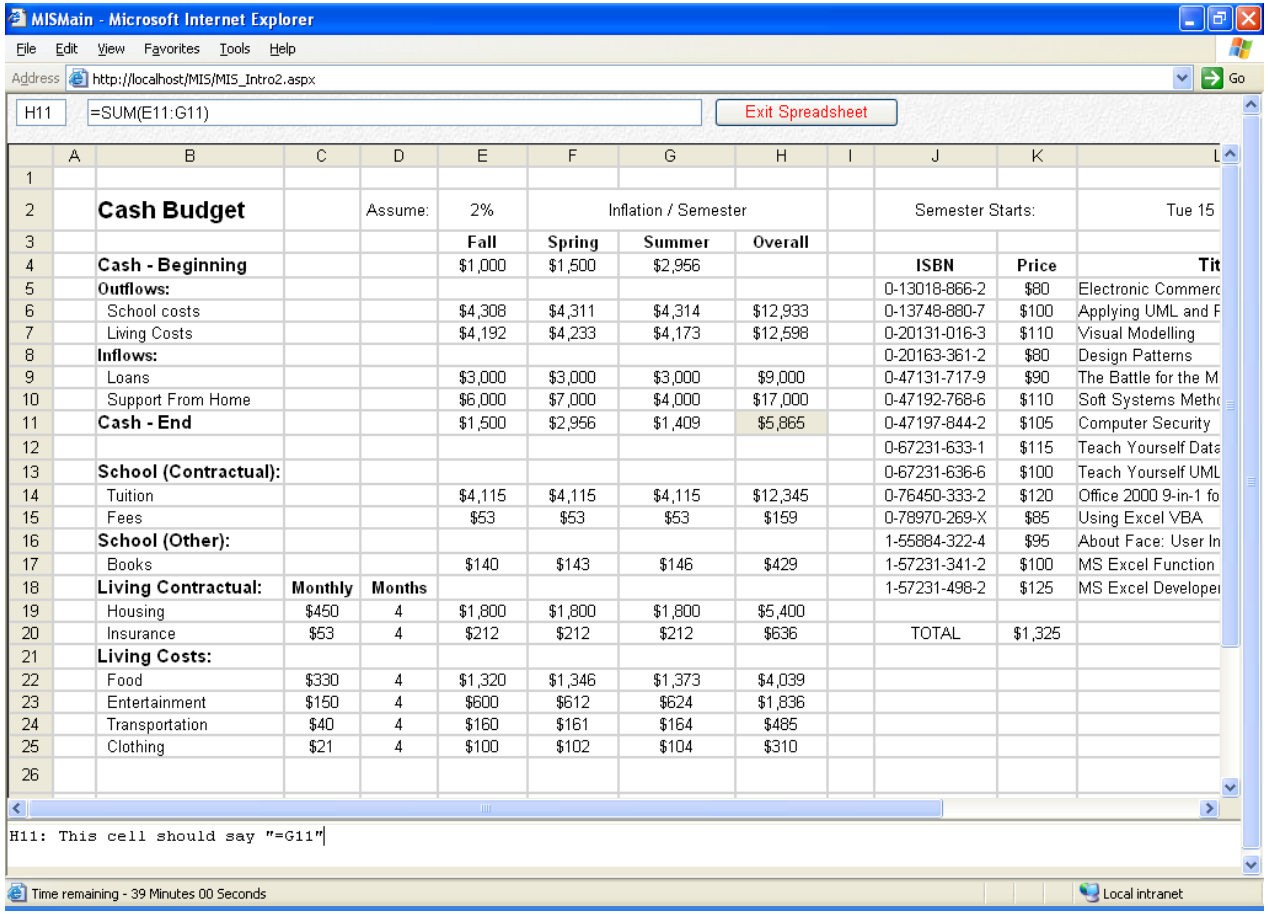
APPENDIX B (cont'd)



APPENDIX B (cont'd)

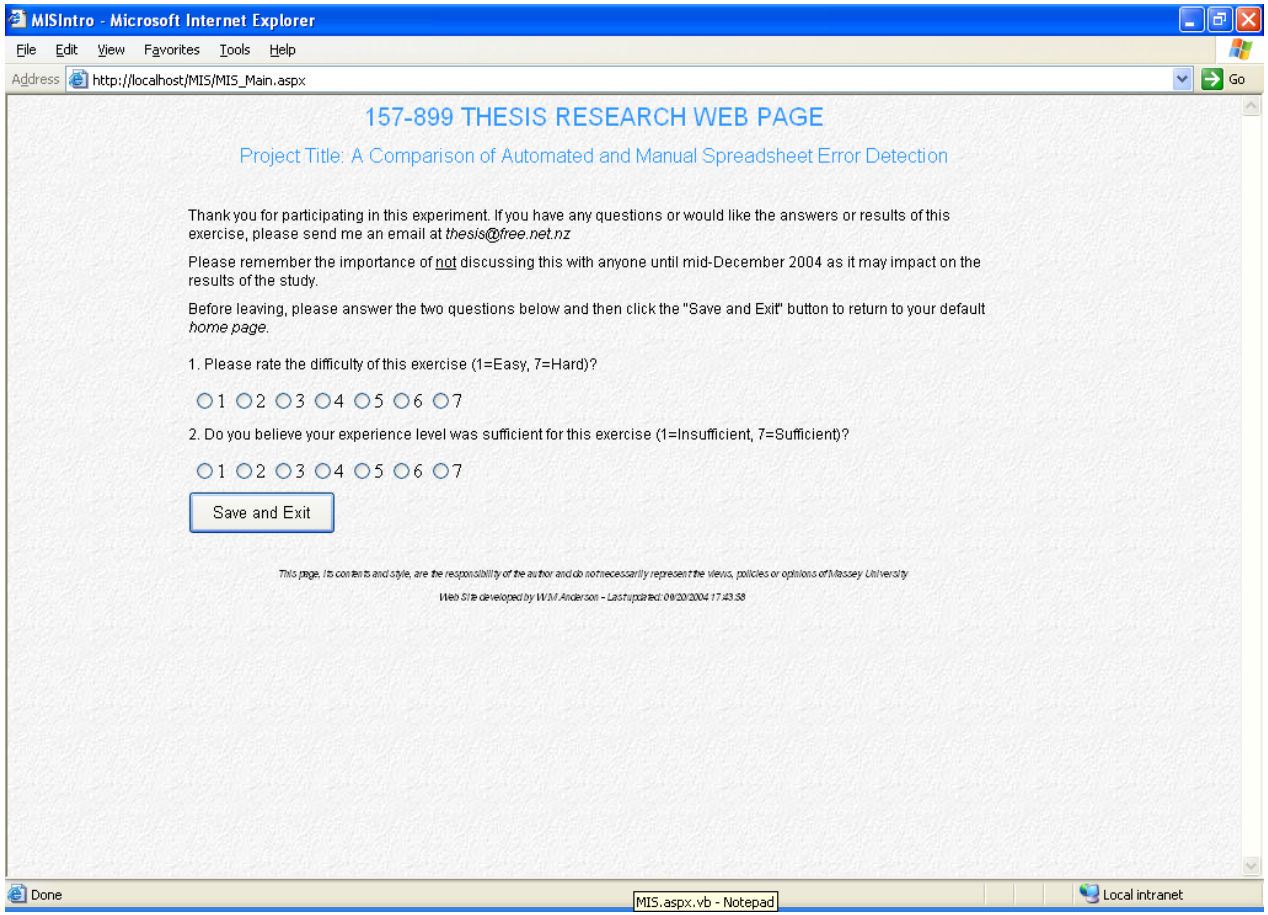


APPENDIX B (cont'd)



	A	B	C	D	E	F	G	H	I	J	K	L
1												
2		**Cash Budget**		Assume:	2%	Inflation / Semester				Semester Starts:		Tue 15
3					**Fall**	**Spring**	**Summer**	**Overall**				
4		**Cash - Beginning**			$1,000	$1,500	$2,956			**ISBN**	**Price**	**Tit**
5		**Outflows:**								0-13018-866-2	$80	Electronic Commerc
6		School costs			$4,308	$4,311	$4,314	$12,933		0-13748-880-7	$100	Applying UML and F
7		Living Costs			$4,192	$4,233	$4,173	$12,598		0-20131-016-3	$110	Visual Modelling
8		**Inflows:**								0-20163-361-2	$80	Design Patterns
9		Loans			$3,000	$3,000	$3,000	$9,000		0-47131-717-9	$90	The Battle for the M
10		Support From Home			$6,000	$7,000	$4,000	$17,000		0-47192-768-6	$110	Soft Systems Methc
11		**Cash - End**			$1,500	$2,956	$1,409	$5,865		0-47197-844-2	$105	Computer Security
12										0-67231-633-1	$115	Teach Yourself Data
13		**School (Contractual):**								0-67231-636-6	$100	Teach Yourself UML
14		Tuition			$4,115	$4,115	$4,115	$12,345		0-76450-333-2	$120	Office 2000 9-in-1 fo
15		Fees			$53	$53	$53	$159		0-78970-269-X	$85	Using Excel VBA
16		**School (Other):**								1-55884-322-4	$95	About Face: User In
17		Books			$140	$143	$146	$429		1-57231-341-2	$100	MS Excel Function
18		**Living Contractual:**	**Monthly**	**Months**						1-57231-498-2	$125	MS Excel Developer
19		Housing	$450	4	$1,800	$1,800	$1,800	$5,400				
20		Insurance	$53	4	$212	$212	$212	$636		TOTAL	$1,325	
21		**Living Costs:**										
22		Food	$330	4	$1,320	$1,346	$1,373	$4,039				
23		Entertainment	$150	4	$600	$612	$624	$1,836				
24		Transportation	$40	4	$160	$161	$164	$485				
25		Clothing	$21	4	$100	$102	$104	$310				
26												

APPENDIX B (cont'd)



APPENDIX C – TEST SPREADSHEET

	A	B	C	D	E	F	G	H	I	J	K	L	M	N
1														
2		**Cash Budget**		Assume:	2%	Inflation / Semester					Semester Starts:	**Tue 15 Mar 04**		
3					**Fall**	**Spring**	**Summer**	**Overall**						
4		**Cash - Beginning**			$1,000	$1,500	$2,956			**ISBN**	**Price**	**Title**	**Group**	
5		**Outflows:**								0-13018-866-2	$80	Electronic Commerce	Architect	
6		School costs			$4,308	$4,311	$4,314	$12,933		0-13748-880-7	$100	Applying UML and Patterns	UML	
7		Living Costs			$4,192	$4,233	$4,173	$12,598		0-20131-016-3	$110	Visual Modelling	UML	
8		**Inflows:**								0-20163-361-2	$80	Design Patterns	Architect	
9		Loans			$3,000	$3,000	$3,000	$9,000		0-47131-717-9	$90	The Battle for the Middle Tier	Architect	
10		Support From Home			$6,000	$7,000	$4,000	$17,000		0-47192-768-6	$110	Soft Systems Methodology	Analysis	
11		**Cash - End**			$1,500	$2,956	$1,409	$5,865		0-47197-844-2	$105	Computer Security	Architect	
12										0-67231-633-1	$115	Teach Yourself Data Structures	Architect	
13		**School (Contractual):**								0-67231-636-6	$100	Teach Yourself UML	UML	
14		Tuition			$4,115	$4,115	$4,115	$12,345		0-76450-333-2	$120	Office 2000 9-in-1 for Dummies	MS Office	
15		Fees			$53	$53	$53	$159		0-78970-269-X	$85	Using Excel VBA	Excel	
16		**School (Other):**								1-55884-322-4	$95	About Face - User Interface Design	GUI	
17		Books			$140	$143	$146	$429		1-57231-341-2	$100	MS Excel Function Reference	Excel	
18		**Living Contractual:**	**Monthly**	**Months**						1-57231-498-2	$125	MS Excel Developers Kit	Excel	
19		Housing	$450	4	$1,800	$1,800	$1,800	$5,400						
20		Insurance	$53	4	$212	$212	$212	$636		TOTAL	$1,325			
21		**Living Costs:**												
22		Food	$330	4	$1,320	$1,346	$1,373	$4,039						
23		Entertainment	$150	4	$600	$612	$624	$1,836						
24		Transportation	$40	4	$160	$161	$164	$485						
25		Clothing	$21	4	$100	$102	$104	$310						
26														
27		**Textbooks To Buy This Year:**					**Entertainment Options:**			**Books In Each Category:**				
28		0-47192-768-6	$110	Soft Systems Methodology			Movies	$8		Architect	5			
29		0-67231-633-1	$115	Teach Yourself Data Structures			Bowling	$8		UML	3			
30		0-67231-636-6	$100	Teach Yourself UML			Bike Riding	$4		Microsoft	3			
31		1-56884-322-4	$95	About Face - User Interface Design			Walking	$2		Analysis	1			
32		1-57231-341-2	$100	MS Excel Function Reference			Party	$200		GUI	1			
33		TOTAL	$520				AVERAGE	$62		TOTAL	13			
34														

APPENDIX C (cont'd)

$B$2	Cash Budget	$C$29	=VLOOKUP(B29,$J$5:$L$18,2)	$F$6	=SUM(F14:F17)
$B$4	Cash - Beginning	$C$30	=VLOOKUP(B30,$J$5:$L$18,2)	$F$7	=SUM(F19:F25)
$B$5	Outflows:	$C$31	=VLOOKUP(B31,$J$5:$L$18,2)	$F$9	3000
$B$6	School costs	$C$32	=VLOOKUP(B32,$J$5:$L$18,2)	$F$10	=ROUND(1000+F6+F7-F9,-3)
$B$7	Living Costs	$C$33	=SUM(C28:C32)	$F$11	=F4-F6-F7+F9+F10
$B$8	Inflows:	$D$2	Assume:	$F$14	4115
$B$9	Loans	$D$18	Months	$F$15	53
$B$10	Support From Home	$D$19	4	$F$17	=ROUND(E17*(1+$E$2),0)
$B$11	Cash - End	$D$20	4	$F$19	=E19
$B$13	School (Contractual):	$D$22	4	$F$20	=E20
$B$14	Tuition	$D$23	4	$F$22	=ROUND(E22*(1+$E$2),0)
$B$15	Fees	$D$24	4	$F$23	=ROUND(E23*(1+$E$2),0)
$B$16	School (Other):	$D$25	4	$F$24	=ROUND(E24+(1+$E$2),0)
$B$17	Books	$D$28	=VLOOKUP(B28,$J$5:$L$18,3)	$F$25	=ROUND(E25*(1+$E$2),0)
$B$18	Living Contractual:	$D$29	=VLOOKUP(B29,$J$5:$L$18,3)	$G$3	Summer
$B$19	Housing	$D$30	=VLOOKUP(B30,$J$5:$L$18,3)	$G$4	=F11
$B$20	Insurance	$D$31	=VLOOKUP(B31,$J$5:$L$18,3)	$G$6	=SUM(G14:G17)
$B$21	Living Costs:	$D$32	=VLOOKUP(B32,$J$5:$L$18,3)	$G$7	=SUM(G19:G24)
$B$22	Food	$E$2	0.02	$G$9	3000
$B$23	Entertainment	$E$3	Fall	$G$10	=ROUND(1000+G6+G7-G4-G9,-3)
$B$24	Transportation	$E$4	1000	$G$11	=G4-G6-F7+G9+G10
$B$25	Clothing	$E$6	=SUM(E14:E17)	$G$14	4115
$B$27	Textbooks To Buy This Year:	$E$7	=SUM(E19:E25)	$G$15	53
$B$28	0-47192-768-6	$E$9	3000	$G$17	=ROUND(F17*(1+$E$2),0)
$B$29	0-67231-633-1	$E$10	=ROUND(1000+E6+E7-E4-E9,-3)	$G$19	=E19
$B$30	0-67231-636-6	$E$11	=E4-E6-E7+E9+E10	$G$20	=E20
$B$31	1-56884-322-4	$E$14	4115	$G$22	=ROUND(F22*(1+$E$2),0)
$B$32	1-57231-341-2	$E$15	53	$G$23	=ROUND(F23*(1+$E$2),0)
$B$33	TOTAL	$E$17	=SUM(C28:C31)/3	$G$24	=ROUND(F24*(1+$E$2),0)
$C$18	Monthly	$E$19	=D19*C19$E$20 =D20*C20	$G$25	=ROUND(F25*(1+$E$2),0)
$C$19	450	$E$22	=D22*C22	$G$27	Entertainment Options:
$C$20	53	$E$23	=D23*C23	$G$28	Movies
$C$22	=11*30	$E$24	=D24*C24	$G$29	Bowling
$C$23	150	$E$25	=D25*25	$G$30	Bike Riding
$C$24	40	$F$2	Inflation / Semester	$G$31	Walking
$C$25	21	$F$3	Spring	$G$32	Party
$C$28	=VLOOKUP(B28,$J$5:$L$18,2)	$F$4	=E11	$G$33	AVERAGE

APPENDIX C (cont'd)

$H$3	Overall
$H$6	=SUM(E6:G6)
$H$7	=SUM(E7:G7)
$H$9	=SUM(E9:G9)
$H$10	=SUM(E10:G10)
$H$11	=SUM(E11:G11)
$H$14	=SUM(E14:G14)
$H$15	=SUM(E15:G15)
$H$17	=SUM(E17:G17)
$H$19	=SUM(E19:G19)
$H$20	=SUM(E20:G20)
$H$22	=SUM(E22:G22)
$H$23	=SUM(E23:G23)
$H$24	=SUM(E24:G24)
$H$25	=SUM(D25:G25)
$H$28	8
$H$29	8
$H$30	4
$H$31	2
$H$32	200
$H$33	=H28+H29+H30+H31+H32/COUNTA(H28:H32)
$J$2	Semester Starts:
$J$4	ISBN
$J$5	0-13018-866-2
$J$6	0-13748-880-7
$J$7	0-20131-016-3
$J$8	0-20163-361-2
$J$9	0-47131-717-9
$J$10	0-47192-768-6
$J$11	0-47197-844-2
$J$12	0-67231-633-1
$J$13	0-67231-636-6
$J$14	0-76450-333-2
$J$15	0-78970-269-X
$J$16	1-55884-322-4
$J$17	1-57231-341-2
$J$18	1-57231-498-2
$J$20	TOTAL
$J$27	Books In Each Category:
$J$28	Architect
$J$29	UML
$J$30	Microsoft
$J$31	Analysis
$J$32	GUI
$J$33	TOTAL
$K$4	Price
$K$5	80
$K$6	100
$K$7	110
$K$8	80
$K$9	$90
$K$10	110
$K$11	105
$K$12	115
$K$13	100
$K$14	120
$K$15	85
$K$16	95
$K$17	100
$K$18	125
$K$20	=SUM(K5:K18)
$K$28	=COUNTIF(M5:M18,"="&J28)
$K$29	=COUNTIF(M5:M18,"="&J29)
$K$30	=COUNTIF(M5:M18,"="&"Excel") +COUNTIF(M5:M18,"="&"MSOffice")
$K$31	=COUNTIF(M5:M18,"="&J31)
$K$32	=COUNTIF(M5:M18,"="&J31)
$K$33	=SUM(K28:K32)
$L$2	=DATE(4,3,15)
$L$4	Title
$L$5	Electronic Commerce
$L$6	Applying UML and Patterns
$L$7	Visual Modelling
$L$8	Design Patterns
$L$9	The Battle for the Middle Tier
$L$10	Soft Systems Methodology
$L$11	Computer Security
$L$12	Teach Yourself Data Structures
$L$13	Teach Yourself UML
$L$14	Office 2000 9-in-1 for Dummies
$L$15	Using Excel VBA
$L$16	About Face - User Interface Design
$L$17	MS Excel Function Reference
$L$18	MS Excel Developers Kit
$M$4	Group
$M$5	Architect
$M$6	UML
$M$7	UML
$M$8	Architect
$M$9	Architect
$M$10	Analysis
$M$11	Architect
$M$12	Architect
$M$13	UML
$M$14	MS Office
$M$15	Excel
$M$16	GUI
$M$17	Excel
$M$18	Excel

APPENDIX D – INVITATION SHEET

THESIS EXPERIMENT

Would you be willing to participate in an experiment for the thesis stage of the Master of Information Systems (MIS) degree I have been doing through Massey University?
If so, please read further and if not - thanks for your time.

What is the thesis about?
The title for the thesis is "*A comparison of manual and automated spreadsheet error detection*". This experiment aims to investigate how well *human participants* and *spreadsheet auditing software* perform when given the task of finding spreadsheet errors and to ascertain whether *human participants* and *spreadsheet auditing software* tend to find different types of errors.

Who is conducting the study?
I am the only student doing this particular study and am being supervised by Peter Blakey, Senior Lecturer in the Department of Information Systems, Massey University Palmerston North.

Whom is the study testing?
This experiment is aimed at Excel users of all skill levels. So everyone who knows how to use Excel is welcome to participate.

How much time is involved?
The time involved is somewhere between 10 and 40 minutes. Experienced users should find they can complete the experiment at the lower end of this timeframe. Once you access the *spreadsheet page*, a timer starts which ensures that you do not remain there longer than 40 minutes.

If you believe you have found all the errors before that time, you can exit early by clicking the "Exit Spreadsheet" button that enables itself automatically after 10 minutes.

What will happen to the information?
The results of this study will be aggregated to ensure the anonymity of participants. No names, email addresses or other identifying information will be reported and only aggregated data will be used for publication. A summary of the results will be sent to interested participants and the source data will be held securely for a period of time once the study is completed.

What do I need to do?
If you believe you know how to use Microsoft Excel and are willing to assist with this study, please reply to this e-mail and let me know that you are happy to participate or would like more information, send me an email at warrena@free.net.nz or phone me on 09-4151550. I will then send you an e-mail containing the URL of the web site and a pass-code that will allow you to gain access to my web site. The site requires Microsoft Internet Explorer 5.0 and above to function correctly.

Please note that the site *emulates* Microsoft Excel, so will behave slightly differently from Excel due to the limitations of JavaScript and HTML.

Finally, if you know of anyone else who may we willing to participate, please feel free to give a copy of this flyer to him or her.

Thank you
Warren M Anderson

APPENDIX E – SCENARIO SHEET

This study is based on the following scenario:

One of your friends, a fellow lecturer, has prepared a simple, one-screen spreadsheet to assist incoming students to do cash budgeting for the three terms. Many of your peers are very concerned that your friend is not experienced enough with spreadsheet formula to make it accurate, and you have been asked to check it over for errors. Because hundreds of students are about to use it, it must be highly accurate.

Even though you are already extremely busy with other tasks, you agree to do it...

The idea of the spreadsheet is that each user would plug in values for items of expense (like books, fees, tuition, etc.), and then for sources of funds (like cash at the beginning of the program and support from parents). The function that might be unfamiliar to you is the ROUND function, which rounds off the needed cash. You can assume that the formula is correct; the -3 parameter specifies the 3rd position before the decimal point.

Other assumptions are as follows:

1. *Beginning cash: $1,000.*
2. *Outflows are composed of contractual and non-contractual school and living costs.*
3. *Inflows take the form of loans and support from home.*
4. *Support from home must be rounded to the nearest $1,000 to ensure that remaining cash remains in the $500 to $1,500 range.*
5. *Non-contractual costs are subject to 2% inflation per semester (and the inflation index can be adjusted to experiment with a variety of scenarios).*
6. *The food cost is averaged. Some meals will be less expensive, and others will be more expensive. The cost for the fall semester takes into account a $2 breakfast, a $4 lunch, and a $6 dinner.*
7. *While looking at a formula in the formula bar area, you can assume that the resultant number matches the formula.*
8. *If you find an error, and it carries over to other cells, only consider the first cell in which the error occurs.*

Now that you know the context of this experiment, there are some things you need to know:

- Please note that the errors you are trying to locate are "formula errors". They are the types of errors that make the spreadsheet produce the wrong results (e.g. formula "=A1+B1+C1" should be "=A1+B1+Z1"). While you may believe that parts of the spreadsheet are incorrect for the scenario described (e.g. it's impractical to buy $500 worth of books in semester one), that is not the sort of error we are looking for.
- It is important that once you have accessed this experiment that you do not discuss the details of the experiment with anyone. In particular, the number, location and types of errors should not be discussed, as doing so could invalidate the results of the study.
- The web site will only remain active until the end of May 2004.
- Once you have started the experiment you must continue until it is completed. Pausing in the middle of the experiment risks having the web page time out by itself, making the results inconclusive.
- You will know how much time is remaining by looking at the status bar on the spreadsheet web page.
- It may be advisable to print this e-mail before starting the experiment so that you can refer to it.

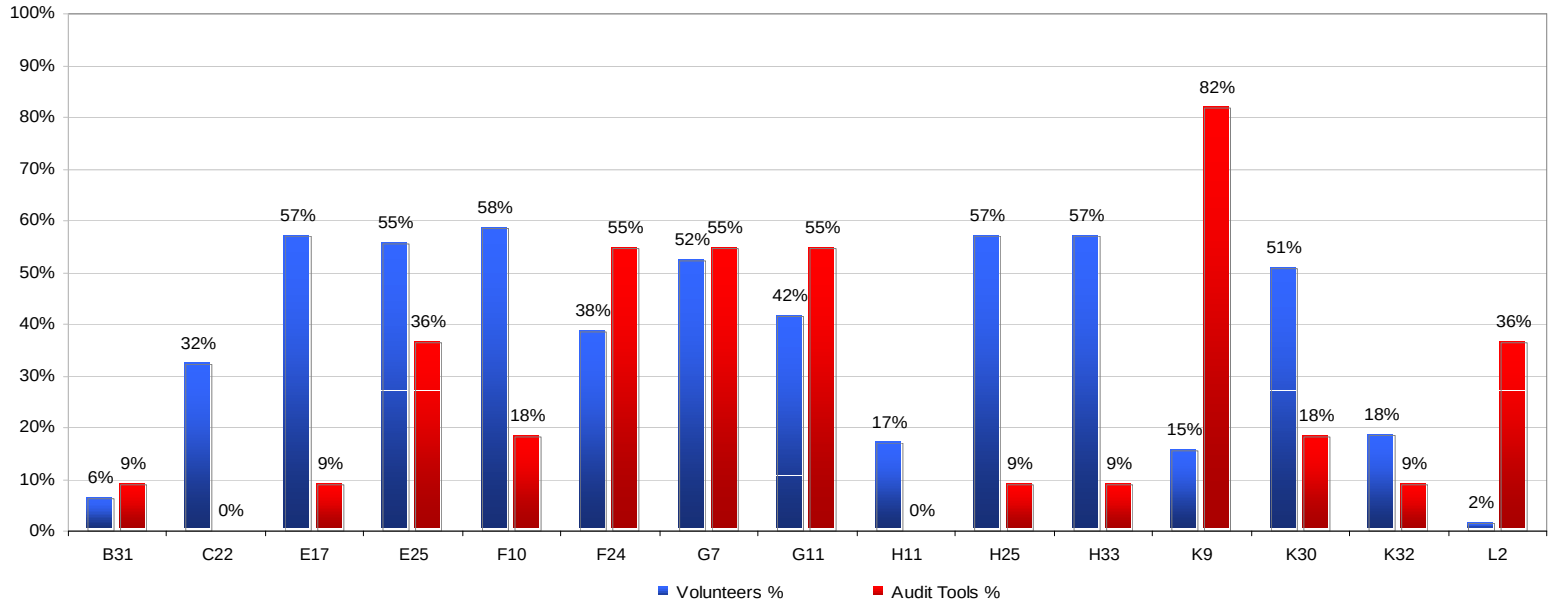
Finally:

The web site can be found at http://thesis.ignition.net.nz/mis.aspx and you will need to enter the ID "«ID»" to be able to access the site.

Thank you again for assisting....

Warren M Anderson

APPENDIX F – ERRORS FOUND



	B31	C22	E17	E25	F10	F24	G7	G11	H11	H25	H33	K9	K30	K32	L2
Participants TRUE	4	21	37	36	38	25	34	27	11	37	37	10	33	12	1
Participants FALSE	61	44	28	29	27	40	31	38	54	28	28	55	32	53	64
Participants %	6.15%	32.31%	56.92%	55.38%	58.46%	38.46%	52.31%	41.54%	16.92%	56.92%	56.92%	15.38%	50.77%	18.46%	1.54%
Audit Tools TRUE	1	0	1	4	2	6	6	6	0	1	1	9	2	1	4
Audit Tools FALSE	10	11	10	7	9	5	5	5	11	10	10	2	9	10	7
Audit Tools %	9.09%	0.00%	9.09%	36.36%	18.18%	54.55%	54.55%	54.55%	0.00%	9.09%	9.09%	81.82%	18.18%	9.09%	36.36%
Find Ranking	14	10	2	5	1	9	6	8	12	2	2	13	7	11	15

REFERENCES

Ahmad, Y., Antoniu, T., Goldwater, S., & Krishnamurthi, S. (2003). *A type system for statically detecting spreadsheet errors.* Paper presented at the Automated Software Engineering, 2003. Proceedings. 18th IEEE International Conference on.

Alavi, M., Nelson, R. R., & Weiss, I. R. (1985). Managing the Risks Associated with End-User Computing. In R. R. Nelson (Ed.), *End-User Computing: Concepts, Issues and Applications* (pp. 230-248). Toronto, Canada: John Wiley & Sons, Inc.

Alavi, M., Nelson, R. R., & Weiss, I. R. (1987). Strategies for End-User Computing: An Integrative Framework. In R. R. Nelson (Ed.), *End-User Computing: Concepts, Issues and Applications* (pp. 271-294). Toronto, Canada: John Wiley & Sons, Inc.

Ayalew, Y., Clermont, M., & Mittermeir, R. (2000). *Detecting Errors In Spreadsheets.* Paper presented at the EuSpRIG 2000 Symposium: Spreadsheet Risks, Audit and Development Methods, University of Greenwich, London.

Babbitt, T. G., Galletta, D. F., & Lopes, A. B. (1998). *Influencing the success of spreadsheet development by novice users.* Paper presented at the International conference on Information systems, Helsinki, Finland.

Ballinger, D., Biddle, R., & Noble, J. (2003a). *Spreadsheet structure inspection using low level access and visualisation.* Paper presented at the Fourth Australian user interface conference on User interfaces, Adelaide, Australia.

Ballinger, D., Biddle, R., & Noble, J. (2003b). *Spreadsheet visualisation to improve end-user understanding.* Paper presented at the Australian symposium on Information visualisation, Adelaide, Australia.

BBC. (2004). *Nasa model 'had wrong data'*. Retrieved July 7, 2004, from http://news.bbc.co.uk/1/hi/world/americas/2930821.stm

Beckwith, L., Burnett, M., & Cook, C. (2002). *Reasoning about Many-to-Many Requirement Relationships in Spreadsheets.* Paper presented at the IEEE Symposia on Human-Centric Computing Languages and Environments,, Arlington, VA.

Benson, D. H. (1983). A Field Study of End-User Computing: Findings and Issues. In R. R. Nelson (Ed.), *End-User Computing: Concepts, Issues and Applications* (pp. 41-54). Toronto, Canada: John Wiley & Sons, Inc.

Berglas, A., & Hoare, P. (1998, April). Spreading the Risk? Errors, risks and techniques in spreadsheets. *Australian CPA, 68*, 42-45.

Berglas, A., & Hoare, P. (1999). Spreadsheet Errors Risks and Techniques. *Management Accounting:*, *77*(7), 46-47.

Blood, A. T. (2002). *Spreadsheet Design Notes*. Retrieved July 15, 2004, from www.XL-Logic.com

Bradley, H. (2001). Expert Office - Formatting Excel. *PC Plus*, *178*, 182.

Bradley, H. (2003). Error-free Excel. *Macworld*, *20*(3), 78-79.

Brandellf, M. (1999, August 2). PC Software TransformsThe PC; VisiCalc's release ignites an explosion in the popularity of microcomputers. *Computerworld*, 62.

Brown, P. S., & Gould, J. D. (1987). An experimental study of people creating spreadsheets. *ACM Trans. Inf. Syst.*, *5*(3), 258-272.

Burnett, M. (1996). *Forms/3 Visual Programming Language Web Site*. Retrieved September 7, 2004, from http://web.engr.oregonstate.edu/~burnett/Forms3/forms3.html

Burnett, M., Cook, C., Pendse, O., Rothermel, G., Summet, J., & Wallace, C. (2003). *End-user software engineering with assertions in the spreadsheet paradigm.* Paper presented at the 25th International Conference on Software Engineering, Portland, Oregon, U.S.A.

Burnett, M., Sheretov, A., & Rothermel, G. (1999). *Scaling up a "What you see is what you test" methodology to spreadsheet grids.* Paper presented at the IEEE Symposium on Visual Languages, 1999.

Butler, R. J. (2000). *Is this Spreadsheet a Tax Evader? How H. M. Customs & Excise Tax Test Spreadsheet Applications.* Paper presented at the 33rd Hawaii International Conference on System Sciences, January 4-7, 2000, Maui, Hawaii.

Butler, R. J. (2002). *The Subversive Spreadsheet*. Retrieved September 10, 2003, from http://www.gre.ac.uk/~cd02/EUSPRIG/RayButler1.htm

Callahan, T. (2002). Block That Spreadsheet Error. *Journal of Accountancy, 194*(2), 59-63.

Campbell-Kelly, M., & Aspray, W. (1996). *Computer: a history of the information machine* (1st ed.). New York: BasicBooks: A Division of HarperCollins Publishers Inc.

Cantellops, D., Bonnin, E., & Reid, A. (2004). *Spreadsheet Design and Validation in Multi-User Application for the Chemistry Laboratory*. Retrieved September 10, 2004, from www.labcompliance.com

Ceruzzi, P. E. (2003). *A History of Modern Computing* (Second ed.). Cambridge England: The MIT Press.

Chadwick, D. (2003). *Stop That Subversive Spreadsheet!* Retrieved November 20, 2003, from http://www.eusprig.org/eusprig

Clermont, M. (2003a). *A Scalable Approach to Spreadsheet Visualisation*. Retrieved August 16, 2004, from https://143.205.180.128/Publications/pubfiles/pdffiles/2003-0175-MC.pdf

Clermont, M. (2003b). *Analyzing Large Spreadsheet Programs.* Paper presented at the 10th Working Conference on Reverse Engineering.

Clermont, M., Hanin, C., & Mittermeir, R. (2002). *A Spreadsheet Auditing Tool Evaluated in an Industrial Context.* Paper presented at the 3rd Annual Symposium of the EuSpRIG.

Computing Canada. (1992, June 8). Lotus: taking notes on the first decade: introducing the first 'killer' PC application: Lotus 1-2-3. *Computing Canada, 18.*

Conatser, K. R. (1995, April). Seven ways to fight worksheet mistakes. *PC World, 100,* 25-27.

Conway, D. G., & Ragsdale, C. T. (1997). Modeling optimization problems in the unstructured world of spreadsheets. *Omega: Int. J. Mgmt Sci., 25*(3), 313-322.

Cook, C., Prabhakararao, S., Main, M., Durham, M., Burnett, M., & Rothermel, G. (2004). Software Engineering for End-User Programmers. *CrossTalk: The Journal of Defense Software Engineering*, 20-23.

Coolican, H. (1996). *Introduction to Research Methods and Statistics in Psychology* (2nd ed.). London: Hodder & Stoughton.

Cooper, D. R., & Schindler, P. S. (2001). *Business Research Methods* (7th ed.). New York: Irwin/McGraw-Hill.

Cragg, P. G., & King, M. (1993). Spreadsheet Modelling Abuse: An Opportunity for OR? *Journal of the Operational Research Society, 44*(8), 743-752.

Davis, G. B. (1989). CAUTION: User-Developed Systems Can Be Dangerous to Your Organization. In R. R. Nelson (Ed.), *End-User Computing: Concepts, Issues and Applications* (pp. 209-228). Toronto, Canada: John Wiley & Sons, Inc.

Davis, J. S. (1996). Tools for Spreadsheet Auditing. *International Journal of Human-Computer Studies, 45*, 429-442.

Ettema, H., Janssen, P., & de Swart, J. (2001). *Spreadsheet assurance by "control around" is a viable alternative to the traditional approach*. Retrieved January 28, 2004, from http://www.gre.ac.uk/~cd02/EUSPRIG/2001/Janssen_2001.htm

EuSpRIG. (2004). *Spreadsheet mistakes - news stories*. Retrieved July 8, 2004, from http://www.eusprig.org/stories.htm

EXChecker. (2003). EXChecker 2.3.015. London: Spreadsheet Auditing Ltd.

Fisher, M., Cao, M., Rothermel, G., Cook, C. R., & Burnett, M. M. (2002). *Automated test case generation for spreadsheets.* Paper presented at the Proceedings of the 24th international conference on software engineering, Orlando, Florida, U.S.A.

Freeman, D. (1996). How to Make Spreadsheets Error-Proof. *Journal of Accountancy, 181*(5), 75-79.

Galletta, D. F., Abraham, D., El Louadi, M., Lekse, W., Pollailis, Y. A., & Sampler, J. L. (1993). An Empirical Study of Spreadsheet Error-Finding Performance. *Journal of Accounting, Management, and Information Technology, 3*(2), 79-95.

Galletta, D. F., Hartzel, K. S., Johnson, S., Joseph, J., & Rustagi, S. (1996). *An experimental study of spreadsheet presentation and error detection.* Paper presented at the Twenty-Ninth Hawaii International Conference on System Sciences, 1996,.

Galletta, D. F., Hartzel, K. S., Johnson, S. E., Joseph, J. L., & Rustagi, S. (1997). Spreadsheet presentation and error detection: An experimental study. *Journal of Management Information Systems, 13*(3), 45-63.

Gasteiger, D. (1993, September). Spreadsheet auditing made easy. *PC World, 11*, 22-23.

Godfrey, K. (1995, January 6). *Computing Error at Fidelity's Magellan Fund*. Retrieved July 7, 2004, from http://catless.ncl.ac.uk/Risks

Hammond, L. W. (1982). Management Considerations for an Information Center. In R. R. Nelson (Ed.), *End-User Computing: Concepts, Issues and Applications* (pp. 67-98). Toronto, Canada: John Wiley & Sons, Inc.

Hopkins, W. G. (2000). *A New View of Statistics: T Test and One Way ANOVA*, 2004, from http://www.sportsco.org/resource/stats/ttest.html

Hormann, F. (1999). Getting the OOPS! Out of Spreadsheets. *Journal of Accountancy, 188*, 79-83.

Horowitz, A. S. (2004). *Spreadsheet Overload?* Retrieved June 1, 2004, from http://www.computerworld.com/printthis/2004/0,4814,93292,00.html

Igarashi, T., Mackinlay, J. D., Chang, B. W., & Zellweger, P. T. (1998). *Fluid visualization of spreadsheet structures.* Paper presented at the IEEE Symposium on Visual Languages. Sep 1998, Halifax, Nova Scotia.

Isakowitz, T., Schocken, S., & Lucas, H. C. J. (1995). Toward a logical/physical theory of spreadsheet modeling. *ACM Trans. Inf. Syst., 13*(1), 1-37.

Janvrin, D., & Morris, J. (2000). *Factors Influencing Risks and Outcomes in End-User Development.* Paper presented at the Twenty-Ninth Hawaii International Conference on System Sciences. 3-6 Jan 1996., Wailea, Hawaii.

Jenkins, A. M. (1985). Research Methodologies and MIS Research. In E. Mumford, R. Hirschhein, G. Fitzgerald & T. Wood-Harper (Eds.), *Research Methods in Information Systems* (pp. 103-117). North-Holland: Elsevier Science Publishers B.V.

Kettinger, W. J., & Lee, C. C. (2002). Understanding the IS-user divide in IT innovation. *Communications of the ACM, 45*(2), 79-84.

Knittel, I. (1986). Cell-Mate locks out errors from Lotus spreadsheets. *PC Magazine, 5*(16), 61-62.

Kreie, J., Cronan, T. P., Pendley, J., & Renwick, J. S. (2000). Applications development by end-users: can quality be improved? *Decision Support Systems, 29*(2), 143-152.

Luftman, J. N., Papp, R., & Brier, T. (2002). *Enablers and Inhibitors of Business-IT Alignment*. Retrieved August 10, 2004, from http://www-1.ibm.com/ibm/palisades/abinsight/issues/2002-Sept/article-4.html

Mamis, R. (1999, May 18). The Tool That Really Changed Everything. *Inc.*

Markus, M. L., & Bjørn-Andersen, N. (1987). Power Over Users: Its Exercise By System Professionals. *Communications of the ACM, 30*(6).

Mason, D. (1989). An Empirical Analysis of Spreadsheet Usage: A solution storing up problems? *Journal of Information Technology, 4*(3), 159-163.

McGill, T., Payne, C., Bennett, D., Carter, K., Chong, A., Hornby, G., et al. (2000). *System Quality, User Satisfaction and End User Development.* Paper presented at the 11th Australasian Conference on Information Systems, Brisbane, Australia.

McLean, E. R. (1979). End Users as Application Developers. In R. R. Nelson (Ed.), *End-User Computing: Concepts, Issues and Applications* (pp. 9-22). Toronto, Canada: John Wiley & Sons, Inc.

Microsoft. (2003). *XL2000: Excel Incorrectly Assumes 1900 Is a Leap Year,* August 23, 2004, from http://support.microsoft.com/default.aspx?scid=kb;en-us;214326

Miric, A. (1999). *The hidden risks of spreadsheets and End User Computing*. Retrieved July 30, 2003, from http://www.itweb.co.za/office/kpmg/9908100916.htm

Morrison, M., Morrison, J., Melrose, J., & Wilson, E. V. (2002). A Visual Code Inspection Approach to Reduce Spreadsheet Linking Errors. *Journal of End User Computing, 14*(3), 51-63.

Moström, J. E., & Carr, D. A. (1997). *Programming paradigms and program comprehension by novices* (No. 1997:10). Luleå, Sweden: Department of Computer Science and Centre of Distance-Spanning Technology, Luleå University of Technology.

Nardi, B. A., & Miller, J. R. (1990a). *An ethnographic study of distributed problem solving in spreadsheet development.* Paper presented at the ACM conference on Computer-supported cooperative work table of contents, Los Angeles, California.

Nardi, B. A., & Miller, J. R. (1990b). *The Spreadsheet Interface: A Basis for End User Programming* (No. HPL-90-08): Hewlett-Packard Laboratories.

Nelson, K. Y. (1993, November). Rows and columns revolution; Windows spreadsheets: they sure ain't VisiCalc. *Home Office Computing, 11*, 52.

Nixon, D., & O'Hara, M. (2001). *Spreadsheet Auditing Software.* Paper presented at the EuSpRIG Conference, 2001, Vrije University, Amsterdam, UK.

OAK. (2001). OPERIS ANALYSIS KIT version 2.03. London: Operis Business Engineering Limited.

O'Beirne, P. (2003). *Agile Spreadsheet Development (ASD)*. Retrieved July 7, 2004, from http://www.sysmod.com/agile.htm

Paine, J. (2004). *Spreadsheet Structure Discovery with Logic Programming.* Paper presented at the EuSpRIG 2004, Klagenfurt.

Panko, R. R. (1995a). *Applications Development: Finding Spreadsheet Errors*. Retrieved June 4, 2004, from http://www.informationweek.com/529/29uwfw.htm

Panko, R. R. (1995b). Finding spreadsheet errors. *InformationWeek*, 100.

Panko, R. R. (1996). *Hitting the wall - errors in developing and debugging a "simple" spreadsheet model.* Paper presented at the 29th Hawaii International Conference on System Sciences. Jan 4-7 1996, Maui, Hawaii.

Panko, R. R. (1997a). *Human Error Website*. Retrieved January 5, 2004, from http://www.cba.hawaii.edu/panko/papers/ss/humanerr.htm

Panko, R. R. (1997b). *Spreadsheet Research (SSR) Website*. Retrieved January 5, 2004, from http://www.cba.hawaii.edu/panko/ssr/

Panko, R. R. (1998). What we know about spreadsheet errors. *J. End User Comput., 10*(2), 15-21.

Panko, R. R. (1999). Applying code inspection to spreadsheet testing. *Journal of Management Information Systems, 16*(2), 159-176.

Panko, R. R. (2000a, July). *Spreadsheet Errors: What We Know. What We Think We Can Do.* Paper presented at the Spreadsheet Risk Symposium, Greenwich, England.

Panko, R. R. (2000b). *Two Corpuses of Spreadsheet Errors.* Paper presented at the 33rd Annual Hawaii International Conference on System Sciences.

Panko, R. R., & Halverson, R. P. (1994). *Individual and group spreadsheet design: patterns of errors.* Paper presented at the Twenty-Seventh Hawaii International Conference on System Sciences.

Panko, R. R., & Halverson, R. P. (1996). *Spreadsheets on trial: a survey of research on spreadsheet risks.* Paper presented at the Twenty-Ninth Hawaii International Conference on System Sciences, Maui, Hawaii, January 1996.

Power, D. J. (2002). *A Brief History of Spreadsheets*. Retrieved March 26, 2004, from http://dssresources.com/history/sshistory.html

Pryor, L. (2004, July). *When, why and how to test spreadsheets.* Paper presented at the Eusprig-2004, Klagenfurt University, Klagenfurt, Austria.

Raffensperger, J. F. (2003). New Guidelines for Spreadsheets. *International Journal of Business and Economics, 2003, 2*(2), 141-154.

Rajalingham, K., Chadwick, D., Knight, B., & Dilwyn, E. (2002). *Efficient Methods for Checking Integrity: An Integrated Spreadsheet Engineering Methodology (ISEM).* Retrieved October 2, 2003, from http://www.kamalasen.com/spreadsheets/rajalingham-99a.pdf

Rajalingham, K., Chadwick, D. R., & Knight, B. (2000). *Classification Of Spreadsheet Errors*. Retrieved January 28, 2004, from http://www.gre.ac.uk/~cd02/EUSPRIG/2001/Rajalingham_2000.htm

Randolph, N., Morris, J., & Lee, G. (2002). *A generalised spreadsheet verification methodology.* Paper presented at the Proceedings of the twenty-fifth Australasian conference on Computer Science, Melbourne, Australia.

Read, N., & Batson, J. (1999). *Spreadsheet Modelling Best Practice*. Retrieved September 10, 2003, from http://www.eusprig.org/smbp.pdf

Reichwein, J., Rothermel, G., & Burnett, M. (1999). *Slicing spreadsheets: an integrated methodology for spreadsheet testing and debugging.* Paper presented at the 2nd conference on Domain-specific languages, Austin, Texas, United States.

Ridington, R. W. (1988, January). The changing way we use spreadsheets. *Lotus, 4,* 156.

Ronen, B., Palley, M. A., & Lucas, H. C. (1989). Spreadsheet analysis and Design. *Communications of the ACM, 32*(1), 84-93.

Rose, F. (1989). The changing user-MIS relationship. *Lotus, 5*(12), 110.

Ross, J. A. (1996, September). Spreadsheet Risk: How and Why to Build a Better Spreadsheet. *Havard Business Review, 74,* 10.

Rothermel, K. J., Cook, C. R., Burnett, M. M., Schonfeld, J., Green, T. R. G., & Rothermel, G. (2000, June). *WYSIWYT testing in the spreadsheet paradigm: An empirical evaluation.* Paper presented at the 24th international conference on Software engineering, Limerick, Ireland.

Rothi, J. A., & Yen, D. (1989). System Analysis and Design in End User Developed Applications. *Journal of Information Systems Education. Sep 1989, 2*(1).

Ruthruff, J. R., Phalgune, A., Beckwith, L., Burnett, M., & Cook, C. (2004). *Rewarding Good Behavior: End-User Debugging and Rewards.* Paper presented at the IEEE Symposium on Visual Languages and Human-Centric Computing, Sep 26-29, 2004 (to appear), Rome, Italy.

Saffo, P. (1989). Looking at VisiCalc 10 years later. *Personal Computing, 13*(11), 233-235.

Sammet, J. E. (1981). History of IBM's Technical Contributions to High Level Programming Languages. *IBM Journal of Research and Development, 25*(5), 520.

Seymour, J. (1984). The Corporate Micro - Left Unchecked, Spreadsheets Can Be a What-If Disaster. *PC Week, 1*(33), 37-38.

Simkin, M. G. (1987). How to Validate Spreadsheets. *Journal of Accountancy, 164*(5), 130-138.

Simkin, M. G. (2004). Ferret out spreadsheet errors: use Excel's tools to uncover and correct formula problems. *Journal of Accountancy, 197*(2), 62-67.

SSD. (2003). Spreadsheet Detective. Queensland: Southern Cross Software.

Stone, D. N., Black, R. L., Wolfe, C., Darazsdi, J. J., Elliott, R. W., King, K. G., et al. (1989). Building Structured Spreadsheets. *Journal of Accountancy, 168*(4), 131-142.

Teo, T. S. H., & Joo Eng, L. P. (2001). Effects of error factors and prior incremental practice on spreadsheet error detection: an experimental study. *Omega: Int. J. Mgmt Sci., 29*(5), 445-456.

Walkenbach, J. (1999). *Microsoft Excel 2000 Formulas*. New York: Wiley Publishing Inc.

Walkenbach, J. (2004). *The Spreadsheet Page*, September 9, 2004, from http://j-walk.com/ss/

Whittle, D., & Janvrin, D. (2002). *Do Cell Labels Improve Spreadsheet Outcomes*. Retrieved June 28, 2004, from http://www.bus.iastate.edu/mpiwowar/seminars/cell_label_%20paper.pdf