Communications of the Association for Information Systems

Volume 25 Article 34

11-1-2009

A Review of Spreadsheet Error Reduction Techniques

Simon Thorne
University of Wales Institute Cardiff, sthorne@uwic.ac.uk

Communications of the Association for Information Systems

CAIS

A Review of Spreadsheet Error Reduction Techniques

Simon Thorne

Information Systems Department, University of Wales Institute Cardiff

sthorne@uwic.ac.uk

David Ball

Information Systems Department, University of Wales Institute Cardiff

Abstract:

Academic and practitioner research shows that spreadsheet errors are prevalent in spreadsheet models and that occasionally these errors cause organisations significant financial loss. A considerable body of research literature now exists on spreadsheet errors and methods to reduce the impact of these errors through technical or organisational approaches. This paper critically examines the literature on spreadsheet error reduction methods and suggests areas and directions of research that would benefit the development of the specific spreadsheet research projects and assist in the mitigation of spreadsheet error.

Keywords: end user computing, spreadsheet risks
.

Volume 25, Article 34, pp. 413-424, November 2009

The manuscript was received 3/10/2009 and was with the authors 1 month for 1 revision.

I. INTRODUCTION

End user development (EUD) describes the activity of end users creating end user applications and information systems using end user software. End user software includes, but is not limited to, word processing software, spreadsheet software, database software and presentation software. Of these end user tools, the most prolific is spreadsheet software, as noted by several authors [SERP 2006; Davies 1987; Jenne 1996; Taylor et al., 1998; Panko and Halverson 1996] (see figure 1).

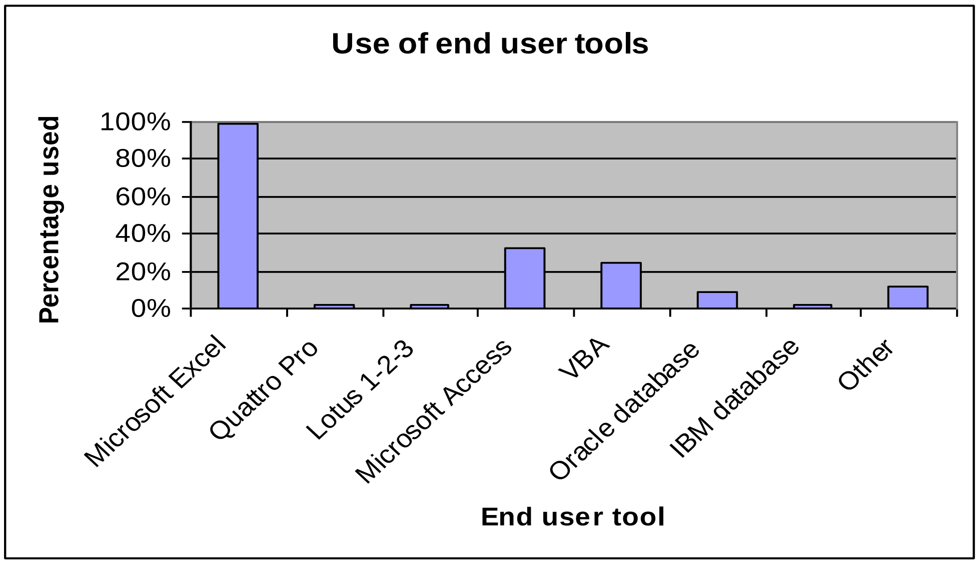

Figure 1. End User Tool Usage [SERP 2006]

Spreadsheet Error

Spreadsheet error is evident in at least 30 percent of all spreadsheet models [Panko 1999]. An example of the impact a spreadsheet error can have in industry was the loss of $24 million by Trans Atlanta Corporation due to a copy and paste error when using a spreadsheet to bid for energy contracts in New York, USA [EuSpRIG 2006]. The loss experienced by the Trans Atlanta Corporation is one example of many where spreadsheet errors have caused significant financial loss in organisations.

If one were to consider smaller organisations, evidence collected from Powell et al., [2009] shows that in 50 spreadsheets from five organizations, the error rates present ranged between one and five percent. The spreadsheets were taken from: two small consulting companies, a large financial firm, a large manufacturing business, and one undisclosed organisation. The impact of these errors is highly variable: in one case, an error caused a cell to be 416.5 percent of its intended value; in another spreadsheet, four separate errors caused the bottom line to be incorrect by over $100,000,000. So it is clear that, large or small, organisations have a spreadsheet error problem.

Reliance on spreadsheet technology for decision support is therefore risky. An example such as Trans Atlanta Corporation is high profile, while smaller organisations escape this level of publicity. It is likely that smaller organisations have lost relatively substantial amounts of money but haven't received the same sort of media attention.

Examples of spreadsheet error research comprise experiments, taxonomies of spreadsheet error, observations of spreadsheet error in practice, technical solutions to reduce spreadsheet error, theories on spreadsheet error management, manual auditing methods and auditing software.

Spreadsheet Error Types

For simplicity, this paper uses the definitions of error types that are most widely cited in spreadsheet literature [Panko and Halverson 1996]. These definitions are not necessarily the most thorough, but are the easiest to apply. Other definitions of spreadsheet error types are neatly summarised in Panko [2009].

Panko and Halverson [1996] split spreadsheet error into quantitative and qualitative types. Within the quantitative error type, Panko and Halverson [1996] discuss three areas of 'known error': mechanical, logical and omission. No detailed explanation is given of the qualitative error type.

Mechanical errors in spreadsheets, according to Panko and Halverson [1996] are:

> *… simple mistakes, such as mistyping a number or pointing to the wrong cell*

From this definition, one can conclude that both mistyping and incorrect cell referencing are mechanical errors. However, it is not clear if syntactical errors are mechanical errors or logic errors, i.e., mistyping the syntax of a command.

Panko and Halverson [1996] define logic error as:

> *… entering the wrong formula because of a mistake in reasoning*

Logic error is therefore based upon the domain knowledge of an individual and the implementation of that knowledge in a spreadsheet. Panko [2005] notes that logic errors are harder to detect and correct than mechanical errors.

Panko and Halverson [1996] define omission error as, *"*when something is left out*"*, and comment that this type of error is the most *dangerous* and the most difficult to detect, a point of view shared by Colver [2007].

Given the definition, omission error can account for anything that is left out of the spreadsheet. This may be the omission of a cell containing a figure in a sum calculation or it may be the omission of a constraint in a rule. This means that omission error has a very broad definition and potentially crosses over with other error types.

Panko and Halverson's definitions of error types are heavily influenced by human error taxonomies, such as Reason [1990] and Allwood [1984].

II. ERROR REDUCTION METHODS

Research on error reduction in spreadsheets is the other distinct research subset that exists in spreadsheet error research.

As discussed below, error reduction research includes lab based auditing, modified software engineering methodologies, and automated tools. Studies of error reduction usually emphasise the effectiveness of an approach, i.e., the effectiveness of a method at preventing, detecting, or reducing error in spreadsheets. This includes subjects as diverse as: manual auditing, software engineering principles, error reduction, or prevention software.

Manual Auditing Methods

Manual auditing is the process of manually checking spreadsheets after creation, using the spreadsheet application and the skill of the auditor. Manual auditing has two separate approaches: individual audit and team audit.

Individual auditing is the process of an individual auditor checking a spreadsheet for mistakes. Research conducted on individual spreadsheet auditing all tends to follow a similar primary experimentation approach. Typically, studies present participants with a spreadsheet seeded with errors which they are required to audit and correct any mistakes found. The researcher then measures the effectiveness of the audit based on the number of mistakes detected and corrected.

Galletta et al. [1993] presented the participants of their study with a model seeded with errors. The participants were asked to analyse the model and find the errors. The study sampled a range of participants with differing spreadsheet development experience. Galletta found, on average, that 56 percent of the seeded errors were discovered. Interestingly the study found that experienced spreadsheet users were quicker at auditing the model but not significantly more accurate. Galletta et al. [1997] extended Galletta et al. [1993] with a larger scale study: the same task was used as the 1993 experiment and a similar detection rate of 51 percent was observed. Panko [1999] conducted a similar auditing experiment to that of Galletta [1993]. Panko found that individual auditors found 63 percent of seeded errors.

Howe and Simkin [2006] offer a similar experiment to that of Galletta [1993] and Panko [1999]. In this study, participants were asked to audit an error seeded spreadsheet. On average, 67 percent of errors were detected. Statistical analysis of demographic information revealed that age, academic ability, and level of education all improved the participant's ability to detect errors.

From the available literature [Galletta et al., 1993; Galletta et al., 1997; Panko 1999; Howe and Simkin 2006], error detection rates using auditing detect approximately 50-60 percent of seeded errors. See Table 1 for a summary of individual audit experiments.

Table 1. Audit Experiments Summary [adapted from Panko 2008]

Study	Subjects	Sample	% errors detected
Galletta *et al*. [1993]	Masters of Business Administration students	60	56%
Galletta *et al.* [1997]	Masters of Business Administration students	113	51%
Panko [1999]	Undergraduates working alone	60	63%
	Undergraduates working in groups of three	60	83%
Howe and Simkin [2006]	Undergraduates	228	67%

The team, or peer, audit differs from individual audits since several auditors work on the same spreadsheet. Researchers suggest that team auditing may have a better error detection rate than individuals (see Table 1). Panko [1999] found that team auditing (groups of three) found 83 percent of errors, as opposed to 63 percent with individual auditors, a finding echoed by Vemula et al. [2006]. Based on this evidence, team auditing appears to be more effective than individual auditing.

The principle of team auditing being superior to individual auditing was recently demonstrated at the EuSpRIG 2007 annual conference. A simple error seeded spreadsheet was given to nine individuals and one group of three to audit. After an hour of auditing, individuals on average found 43 percent of errors and the group of three found 74 percent of errors.

Software Engineering Methods

Researchers have sought to apply software engineering methods to spreadsheets to reduce error [Rajalingham et al., 2000; Burnett et al., 2001; Grossman 2002; Burnett et al., 2003; Yirsaw et al., 2003; Burnett et al., 2004; Grossman and Ozluk 2004; Pryor 2004; Panko 2006].

Considering that the origins of software engineering were the software crisis (i.e., poor quality software), adapting such techniques to spreadsheets is a promising concept.

The focus of software engineering research in spreadsheets varies. Some researchers investigate spreadsheet best practice [Grossman 2002], some investigate structured development in spreadsheets [Burnett *et al.*, 2003] and others examine spreadsheet testing [Panko 2006].

Some researchers [Grossman 2002; Burnett et al., 2004] suggest that spreadsheet error can be managed by adapting software engineering methods to create a special discipline called "spreadsheet engineering".

Spreadsheet Engineering

Grossman [2002] presents eight principles for a spreadsheet engineering methodology:

1. *Best practice can have a large impact*
2. *Lifecycle planning is important*
3. *A priori requirements specification is beneficial*
4. *Predicting future use is important*
5. *Design matters*
6. *Best practice is situation dependent*

7. *Programming is a social, not an individual activity*
8. *Deployment of best practices is difficult and consumes resources*

Grossman discusses challenges in a spreadsheet engineering methodology, identifying the need for a taxonomy of spreadsheet user experience. Grossman [2002] notes that a consensus on practices to improve spreadsheet quality is difficult, since quality in spreadsheet modelling is a subjective issue. This is a notion supported by Colver [2007], who remarks that "best practice*"* is a contentious issue, because the application of best practice approaches often contradict one another.

Rajalingham et al*.* [2000] discuss a structured methodology for spreadsheet modelling using data diagrams and modularisation of spreadsheets. Rajalingham et al*.* [2000] consider traditional approaches to data modelling through the use of entity relationship (ER) diagrams [Chen 1975] and Jackson's structured diagrams (JSD) [Jackson 1975]. The research demonstrated that using JSD, spreadsheets can be modularised, which aids the future maintenance of the model. However, this approach would require spreadsheet modellers to know how to apply JSD and, in practice, would require training.

Burnett et al*.* [2003] discusses end user software engineering in spreadsheets giving "user assertions" (pointers to potential errors) for debugging a spreadsheet. The assertions assisted the spreadsheet modellers to detect and correct spreadsheet errors that would have otherwise been missed. Burnett *et al.* [2004] suggest that, since spreadsheet modellers are not IS professionals, it is more practical to use a small feedback loop rather than a comprehensive SDLC-based approach. The results of their experiment showed that spreadsheet modellers found this new feedback approach easier to put into operation than a strict software engineering approach.

Grossman and Ozluk [2004] extend previous work on spreadsheet engineering principles, Grossman [2002], to give a more traditional adaptation of the SDLC:

1. *Modelling*
2. *Development parameters*
3. *Design*
4. *Programming*
5. *Quality Control*
6. *Debugging*
7. *Documentation*
8. *Usage*
9. *Modification*

This spreadsheet engineering framework takes special consideration of how spreadsheet models are used in practice. In particular, stages 8 (usage) and 9 (modification) acknowledge that the use of a spreadsheet may change and that the spreadsheet may be modified in the future. However, Grossman and Ozluk [2004] provide only the theoretical benefits and they do not include any data to indicate if this approach improves quality in practice.

Rust et al*.* [2006] demonstrate the potential of test driven development (TDD) as a means to develop spreadsheets. They found that by using TDD to develop spreadsheet models, the quality of the resulting spreadsheet model potentially increased. Test driven development is a software development approach based upon writing test cases first and then producing code which satisfies the tests. This approach seems to be particularly suited to spreadsheets since it scales down to the relative simplicity of spreadsheet models. The application of a more traditional software development approach, such as the waterfall model, would be infeasible and inappropriate for spreadsheet models.

In conclusion, spreadsheet engineering is grounded in software engineering principles that can provide theoretical benefits. However, establishing 'spreadsheet engineering' will take more time and research.

Spreadsheet Testing

There are few studies which provide testing strategies for spreadsheets, although testing is identified as an important consideration for reducing error [Panko 2006].

Burnett et al*.* [2001] provides a testing methodology designed to help users locate errors before the model is implemented. They combine a testing methodology with an interactive audit tool which indicates potentially erroneous cells on the spreadsheet.

Pryor [2004] adapts software engineering tests [Pressman and Ince 2000] to synthesize a technique suitable for spreadsheets. Pryor suggests the following tests: Unit testing (individual units as the spreadsheet is developed); System testing (the performance of the spreadsheet as a whole); Regression testing (comparing the spreadsheet with its predecessors); and Acceptance testing (user acceptance of the spreadsheet). Whilst this method is proven in software engineering [Pressman and Ince 2000], Pryor [2004] offers no data to suggest that such an approach is effective in spreadsheets.

Yirsaw *et al.* [2003] describe a software engineering debugging method applied to spreadsheets, which applied a method called 'interval based testing' to spreadsheets. Interval based testing is a 'dynamic' testing method intended to be used as the spreadsheet is being developed. Yirsaw argues that interval based testing allows spreadsheet modellers to detect errors before they are implemented. However, no practical evidence is provided to show that modellers can use this approach or any indication of spreadsheet error detection rate.

Panko [2006] summarises testing techniques for spreadsheets and recommends a strategy for spreadsheet testing. This strategy advocates that testing should account for 25 to 40 percent of all spreadsheet development time. It also suggests that specific testing methodologies, such as the Fagan method [Fagan 1986], should be used on planning documents as well as spreadsheets. The Fagan method is an iterative testing cycle conducted by a team of testers. The Fagan method has been shown to reduce defects by 80 to 90 percent.

Spreadsheet Auditing Software

Research into spreadsheet auditing and testing has led to development of partially automated software tools. These tools appear as software add-ins to the standard spreadsheet application. These tools offer automated auditing functions, spreadsheet control functions, and alternative spreadsheet programming environments. Spreadsheet auditing software is defined as a third party vendor add-in which performs auditing functions.

Spreadsheet auditing software has two basic methods, which are referred to as: standard and specialised. Standard auditing functions include cell dependency tracing, collating and displaying of formulae, and indication of potentially erroneous formulae. Typically, the auditing software provides a number of graphical representations of the spreadsheet, such as a "formula map".

Specialised auditing functions have applications in particular industries, usually in addition to standard functions. For example, Spreadsheet Auditing from Customs and Excise (SpACE), as discussed by Butler [2000], performs a variety of "standard" auditing functions and has "specialised" functions that relate VAT calculations. There are many examples of spreadsheet auditing software, which generally have a similar functionality. However, there are some pieces of software that offer novel features to reduce spreadsheet error.

A novel approach is taken by XLAnalyst. This software offers standard audit functionality but also attempts to quantify spreadsheet risk. The risk calculation is based upon the potential errors found in that spreadsheet. Research shows that when faced with a large volume of spreadsheets, it is infeasible to audit all of them [Nash and Goldberg 2005; Butler 2000; Pryor 2004, 2003].Software such as XLAnalyst, which could prioritise the spreadsheets according to risk, would address auditing issues raised by many authors [Nash and Goldberg 2005; Butler 2000; Pryor 2004; Pryor 2003].

However, measuring risk in spreadsheets is particularly difficult [Madahar et al*.,* 2007].To date, there is no agreed mechanism for measuring risk in spreadsheets. Research on spreadsheet risk management and classification conducted by Madahar et al*.* shows some potential on reaching an acceptable risk classification model. However, this model has yet to be evaluated fully in practice.

Clermont presents a tool for auditing spreadsheets in a series of papers [Clermont and Mittermeir 2002; Clermont 2003; Clermont and Mittermeir 2003; Clermont 2004]. Clermont's software tool kit allows the user to visualise large spreadsheets. The visualisation process converts spreadsheets into hierarchical and graphical based representations. The visualisation is based upon the logical areas of a spreadsheet, the semantic classes, and the data modules. Visualisation allows modellers to spot inconsistencies in data and erroneous values. Clermont and Mittermeir [2002] demonstrate the visualisation tool kit on spreadsheets gathered from industry. The tool kit found 241 material errors in three real-world spreadsheets from a company claiming the spreadsheets were "error free". However, the actual auditing was done by the creators of the tool kit. Therefore, the practical usability of these tools remains unclear. A study into the usability of the kit by spreadsheet practitioners would reinforce the value of this work.

Nixon and O’Hara [2001] underline weaknesses in spreadsheet auditing software by experimenting with auditing packages and an error seeded spreadsheet. The research found that the auditing software could detect close to 80 percent of the errors in the spreadsheet. For example, the auditing software, using Panko and Halverson’s [1996] error types, was found to be strong at detecting mechanical and logic errors but poor at detecting omission errors. Further, Nixon and O’Hara stress that audit software can only indicate where errors potentially lie, the effectiveness of the software is therefore still partially dependant on the skill of the auditor.

Flood and McDaid [2007] present a novel approach to debugging spreadsheets using voice recognition auditing software. However, the results showed that debugging the spreadsheets by voice took almost twice as long and detected 15 percent less errors when compared to traditional spreadsheet auditing methods.

Spreadsheet Control Software

Spreadsheet control software is defined as software that allows the management of spreadsheet models by controlling how spreadsheets are used. There are currently two types of control mechanism; centralised and decentralised.

Centralised control mechanisms, such as Google spreadsheets, require modellers to use or download spreadsheets from a central server, modify them, and then replace them. These systems keep copies of previous versions so that if a mistake is made, a rollback can be performed. Modellers who wish to use or download spreadsheets have to log into a server and all changes to the spreadsheet are recorded providing an audit trail.

Decentralised control systems place controls on the users’ PCs to monitor spreadsheet usage and modification. Typically these systems employ agent technologies that monitor spreadsheet modification and make comparisons between versions of the same spreadsheet. Any changes to the spreadsheet are recorded and attributed to a user, providing an audit trail.

“Telltable” is a centralised control mechanism developed by Nash [Nash 2003; Nash and Goldberg 2005]. The product allows the user to track changes and rollback to previous versions. This software also has standard auditing capabilities. Two observations can be made of centralised control mechanisms; firstly, the need for investment in technology, and secondly, a change in ‘normal’ use of spreadsheets. Investment in technology may be necessary to adopt a centralised system; for example, a suitable application server may need to be purchased. ‘Normal’ use of spreadsheets is defined as spreadsheet software residing on a modellers own PC, which they open and use on their PC. A centralised system requires a hosted or downloadable spreadsheet.

One approach to utilising agent technologies in spreadsheets to reduce spreadsheet error is proposed in Thorne et al. [2003]. Baxter [2004] presents a decentralised control system that uses agents to monitor change in spreadsheets. Agent software monitors spreadsheets on a network and once a change is recorded, two versions of the same spreadsheets are analysed for differences. A report is then generated, identifying changes and by whom those changes were made, providing an audit trail.

The approach of Baxter [2004] does not require the investment in infrastructure or centralisation of spreadsheet software that Nash [2003] and Nash and Goldberg [2005] require. However, it does make use of decentralised agent technology, which attracts criticism. For example, Nwana and Ndumu [1999] note security as a primary concern in agent technologies, whilst other criticisms of the technology include increased loading on LAN bandwidth and unsatisfactory transaction control mechanisms.

Alternative Spreadsheet Programming Environments

Alternative spreadsheet programming environments are defined as non conventional methods for programming spreadsheets. Conventional spreadsheet programming methods are defined as the traditional matrix analogy using cells and formulae, such as Microsoft Excel or Lotus 123.

Thorne and Ball [2009] discuss a novel approach to representing decision support spreadsheet models called Example Driven Modelling (EDM). This technique uses attribute classifications (user generated example input and output) and Neural Networks to create equivalent spreadsheet models. EDM has been shown to be more tolerant to user error than equivalent spreadsheet models and addresses some of the human factor issues identified by Panko [1999].

Paine [2001] presents “Model Master” a tool for backward engineering and creating spreadsheets. Model Master converts spreadsheets into precise mathematical notation, with the output resembling traditional computer programming code. Model Master allows spreadsheet modellers to construct new spreadsheets by coding the spreadsheet in the Model Master language and then converting the language into a spreadsheet. Paine [2005] demonstrates how modularity can be achieved in spreadsheet models via Model Master. Paine [2005] argues that modularity allows the effective management of large spreadsheet models.

In Paine and Williamson [2006] and Paine [2007], Model Master is renamed as “Excelsior”, with new emphasis being placed on the benefits of modularity. However, since Excelsior and Model Master are similar to traditional programming languages and considering that most spreadsheet developers are non-IS professionals, they are unlikely to have the programming experience necessary to make effective use of a such a tool without training.

III. CONCLUSIONS ON ERROR REDUCTION METHODS

The most effective means of reducing errors in spreadsheets based upon the available literature is manual auditing. Individual audits find between 51 and 67 percent of errors, whilst team audits find 83 percent of errors (see Table 1).

The application of software engineering principles to spreadsheets and establishing a ‘spreadsheet engineering’ discipline has the potential to significantly reduce spreadsheet errors. Evidence presented by Rajalingham et al. [2000], Burnett et al. [2001], Burnett et al. [2003], and Burnett et al. [2004], show reductions in errors that are gained by applying software engineering principles to spreadsheets.

Spreadsheet testing conducted by Grossman [2002], Yirsaw et al. [2003], Grossman and Ozluk [2004], Pryor [2004], and Panko [2006] highlight the potential benefits to be gained from software engineering testing methods and the creation of a spreadsheet engineering. However, no data is included in these papers to prove the effectiveness of such approaches.

Auditing software has an unknown impact on reducing spreadsheet errors. Whilst the software is good at finding particular types of error, as Nixon and O’Hara [2001] stress, auditing software can only point to potential errors; deciding if they are errors and how to correct them is left to the auditor.

One exception to these conclusions on auditing software is work conducted by Clermont [Clermont and Mittermeir 2002; Clermont 2003; Clermont and Mittermeir 2003; Clermont 2004]. The work above demonstrates how a visualisation tool can be used to audit spreadsheets in a unique manner. Further, the authors take a number of spreadsheets from industry and audit them using this software. The only criticism of this is that the auditing was conducted by the authors, raising questions of usability by spreadsheet modellers themselves.

Spreadsheet control software, both in centralised and decentralised environments, offers a means of controlling spreadsheet development in an organisation rather than directly reducing errors. These control systems offer theoretical benefits but lack hard evidence that proves the effectiveness of such software. Further, there are potential security and investment disadvantages associated with some of the approaches.

Alternative spreadsheet programming environments [e.g., Paine 2001; Paine 2005; Paine and Williamson 2006; Paine 2007] have the potential to reduce error, but there has not been a substantial field study to see if the approach is effective.

Reflections and Recommendations

There now exists a significant body of published spreadsheet research. However, spreadsheet research as a research discipline lacks of an overall research framework. Without this framework, future research efforts are likely to continue in piecemeal fashion, with research being conducted in isolated avenues with little overarching direction.

A framework for spreadsheet research could be based on either a business driven agenda or a software engineering driven agenda. The business-driven spreadsheet research agenda could be led by risk, i.e., establishing an acceptable system of assessing use and risk for spreadsheet applications and then developing strategies to deal with the identified risk. This risk assessment strategy should range from the business critical applications to the trivial, everyday spreadsheets. Research could then be focussed on meeting those specific risks and challenges, making spreadsheet research a more cohesive pragmatic discipline. However, at present there is no published, agreed means of assessing spreadsheet use and subsequent risk. Some significant progress on spreadsheet use and risk management has been achieved through the work of Madahar et al. [2007], although this method is yet to be fully realised. Clearly, reaching a consensus on use and risk of spreadsheets is difficult but greatly needed.

Another alternative is that a framework from software engineering could be adopted and adapted to create spreadsheet engineering. In fact, the current state of end user spreadsheet applications is akin to that of the “software crisis” in the 1960s when software was developed in an ad hoc manner and without standards. Many of the truths of the software crisis apply to spreadsheet development, such as the lack of formalised methodologies, testing regimes, and education. Due to these similarities, it would seem a logical step to adapt a software engineering framework and principles for spreadsheets. Some researchers have adapted software engineering methods, such as Burnett et al*.* [2001], Clermont and Mittermeir [2002], and Rust et al*.* [2006]. Attempts to adapt a software engineering framework to spreadsheet engineering, such as Grossman and Ozluk [2004], are a good starting point but need further development.

Alternatively, one could try to accumulate the heuristic "best practice" knowledge from current spreadsheet literature and use that as a basis for organising existing work and directing future research. At face value, this seems a potentially productive direction in which to head; especially considering that most spreadsheet developers are not software engineers or risk management professionals and are more likely to be able to contribute on the "best practice" level. However, because spreadsheets are used in such a variety of ways in a variety of industries, finding common ground amongst them is difficult. In fact, as Colver [2007] notes, the application of best practice in one context often has a negative consequence in another.

An emerging area of spreadsheet research that is a significant source of spreadsheet error is human factors and human error. It has been suggested by several authors [Panko 2007; Thorne and Ball 2008] that spreadsheet errors may be more grounded in human error than any other discipline. Human Factors describes a range of effects such as base error rate, overconfidence, and bias, which all can have an impact on spreadsheet quality. Human Factors has its own set of engineering principles that may prove to be beneficial when adapted to spreadsheet error mitigation.

To address spreadsheet errors in business, the answer is, perhaps, to further develop spreadsheet engineering so that it is well focused on real world business problems and is easily applicable by end users. Evidence suggests that a business driven approach is more effective [Kottemann et al*.,* 2009] and, considering that the large majority of spreadsheet modellers are not formally trained [Pemberton and Robson 2000], making the techniques easy to understand and apply is paramount. Of course, before any of this can be considered the business world must be aware of the errors, be able to identify problematic areas of operation, and then apply techniques to reduce the likelihood or impact of spreadsheet errors.

REFERENCES

Editor's Note: The following reference list contains hyperlinks to World Wide Web pages. Readers who have the ability to access the Web directly from their word processor or are reading the paper on the Web can gain direct access to these linked references. Readers are warned, however, that:

1. These links existed as of the date of publication but are not guaranteed to be working thereafter.
2. The contents of Web pages may change over time. Where version information is provided in the References, different versions may not contain the information or the conclusions referenced.
3. The author(s) of the Web pages, not AIS, is (are) responsible for the accuracy of their content.
4. The author(s) of this article, not AIS, is (are) responsible for the accuracy of the URL and version information.

Allwood, C. (1984). "Error Detection Processes in Statistical Problem Solving," *Cognitive Science* 8, pp. 413-437.

Baxter, R. (2004). "End User Computing Applications, Auditability and Other Benefits Derived from a Temporal Dimension," *Proceedings of EUSPRIG 2004 – Risk reduction in End User Computing -'Best Practice for Spreadsheet Users in New Europe'* Klagenfurt, Austria, pp. 67-70, 1-902724-94-1.

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

Burnett, M., C. Cook., P. Pendse., G. Rothermel, J. Summet, And C. Wallace. C (2003). "End User Software Engineering with Assertions in the Spreadsheet Paradigm," *Proceedings of International Conference on Software Engineering* May 2003, Corvallis, Oregon, pp. 33-59.

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

Butler, R. (2000). "Risk Assessment for Spreadsheet Developments: Choosing which Models to Audit," *EuSpRIG 2000 Symposium, Spreadsheet Risks—the Hidden Corporate Gamble* Greenwich, London pp. 47-56.

Chen, P (1975). "The Entity Relationship Model – Towards a Unified View Of Data," *Transactions of the ACM* 17(4), pp. 9-36.

Clermont, M. and R. Mittermeier (2002). "Finding High Level Structures in Spreadsheet Programs," *9th Working Conference on Reverse Engineering* (WCRE 2002) pp. 221-232.

Clermont, M. and R. Mittimeir (2003). "A Pattern Based Approach to Spreadsheet Auditing," *Proceedings of the 6th International Conference on Information Systems Implementation and Modelling* Frankfurt, Germany.

Clermont, M. (2003). *A Scalable Approach to Spreadsheet Visualisation* PhD thesis, available from Universitaet Klagenfurt, Austria.

Clermont, M. (2004). "A Toolkit for Scalable Spreadsheet Visualisation," *Proceedings of EUSPRIG 2004 – Risk reduction in End User Computing - Best Practice for Spreadsheet Users in New Europe* Klagenfurt, Austria, pp. 45-52, 1-902724-94-1

Colver, D. (2007). "Inclusion Analysis: Finding Omission Errors," *Proceedings of EuSpRIG 2007 – Enterprise Spreadsheet Management: A Necessary Evil?* Greenwich, London, ISBN 978-905617-58-6.

Davis, G. (1987). "Commentary on Information systems," *Accounting Horizons* 43, pp. 103 -105.

EuSpRIG (2006). "Spreadsheet Horror Stories," http://www.eusprig.org/stories.htm, (accessed June 23, 2006).

Fagan, M. E. (1986). "Advances in Software Inspections," *IEEE Transactions on Software Engineering* 12(7), pp. 744-751.

Flood D. and K. McDaid (2007). "Voice Controlled Debugging of Spreadsheets," *Proceedings of EuSpRIG 2007 – Enterprise Spreadsheet Management: A Necessary Evil?* Greenwich, London, pp. 155-165, ISBN 978-905617-58-6.

Galletta, D. F., K. S. Hartzel, S. Johnson, and J. L. Joseph (1997). "Spreadsheet Presentation and Error Detection: An Experimental Study," *Journal of Management Information Systems* 13 (2), pp. 45-63.

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

Grossman, T. and O. Ozluk (2004). "A Paradigm for Spreadsheet Engineering Methodologies," *Proceedings of EUSPRIG 2004 – Risk Reduction in End User Computing - Best Practice for Spreadsheet Users in New Europe* Klagenfurt, Austria, pp. 13-24, 1-902724-94-1.

Grossman T. (2002). "Spreadsheet Engineering: A Research Framework," *Proceedings of European Spreadsheet Risks Interest Group, 3rd Annual Symposium* Cardiff, UK pp. 21-34, ISBN 1861661827.

Howe, H. and M. Simkin (2006), "Factors Affecting the Ability to Detect Spreadsheet Errors," *Decision Sciences Journal of Innovative Education* 4(1), pp. 101-122.

Jackson, M. A. (1975). *"Principles of Program Design,"* Academic Press.

Jenne, S. (1996). "Audits of End User Computing," *Internal Auditor* 53(6), pp. 30-35.

Kotteman, J., K. Boyer-Wright, J. Kincaid, and F. Davis (2009). "Understanding Decision-Support Effectiveness: A Computer Simulation Approach," *IEEE Transactions on Systems, Man and Cybernetics – Part A: Systems and Humans* 39(1), pp. 57-65.

Madahar, M., P. Cleary, and D. Ball (2007). "Categorisation of Spreadsheet use within Organisations: A Progress Report," *Proceedings of EuSpRIG 2007 – Enterprise Spreadsheet Management: A Necessary Evil?* Greenwich, London, pp. 37-47, ISBN 978-905617-58.

Nash, J. and J. Goldberg (2005). "Why, How and When Spreadsheet Tests should be Used," *Proceedings of EUSPRIG 2005 – Managing Spreadsheets in the Light of Sarbanes-Oxley* London, UK, pp. 82-94, ISBN 1-902724-16-X.

Nash, J. (2003). "Audit Change Analysis of Spreadsheets," *Proceedings of EUSPRIG 2003 – Building Better Spreadsheets – 'From the Ad-Hoc to the Quality Engineered'* Dublin, Ireland, pp. 81-91, ISBN 1-86166-199-1.

Nixon, D. and M. O'Hara (2001). "Spreadsheet Auditing Software," *Proceedings of EUSPRIG 2001 – Spreadsheet Risks, Audit and Development Models* pp. 79-95, ISBN 1861661797.

Nwana, H. and Ndumu (1999). "A Perspective on Software Agent Research," *The Knowledge Engineering Review* 14, pp. 125-142.

Paine, J. (2001). "Ensuring Spreadsheet Integrity with Model Master," *Proceedings of EUSPRIG 2001 – Spreadsheet Risks, Audit and Development Models* Amsterdam, Netherlands, pp. 17-39, ISBN 1861661797.

Paine, J. (2005). "Excelsior: Bringing the Benefits of Modularity to Excel," *Proceedings of EUSPRIG 2005 – Managing Spreadsheets in the Light of Sarbanes-Oxley* Greenwich, London, pp. 161 - 173, ISBN 1-902724-16-X.

Paine, J. (2007). "Practical Experience with Excelsior," *Proceedings of EuSpRIG 2007 – Enterprise Spreadsheet Management: A Necessary Evil?* Greenwich, London, pp. 131-143, ISBN 978-905617-58-6.

Paine, J., E. Tek, and D. Williamson (2006). "Rapid Spreadsheet Reshaping with Excelsior: Multiple Drastic Changes to Content and Layout are Easy when you Represent Enough Structure," *Proceedings of Eusprig 2006 – Managing Spreadsheets: Improving Corporate Performance, Compliance and Governance* Greenwich, London, pp. 129 – 147, ISBN 1-905617-08-9.

Panko, R. and R. Halverson (1996). "Spreadsheets on Trial: A Survey of Research on Spreadsheet Risks," *Hawaii International Conference on System Sciences* Maui, Hawaii, pp. 326-335.

Panko, R. (1999). "What we Know about Spreadsheet Errors," *Journal of End User Computing, Special Issue: Scaling up End User Development*, pp. 15-22.

Panko, R. (2006). "Recommended Practices for Spreadsheet Testing," *Proceedings of EuSpRIG 2006 – Managing Spreadsheets: Improving Corporate Performance, Compliance and Governance* Greenwich, London, pp. 73-85, ISBN 1-905617-08-9.

Panko, R. (2008). *"Spreadsheet Error Rates,"* http://panko.shidler.hawaii.edu/SSR/index.htm (accessed October 6, 2008).

Panko, R. (2009). "Revisiting the Panko-Halverson Taxonomy of Errors," *42nd Hawaii International Conference on Systems Science* Kona, Hawaii.

Pemberton, J. and A. Robson (2000). "Spreadsheets in Business," *Industrial and Management and Systems* 100(8), pp. 379-388.

Powell, S., K. Baker, and B. Lawson (2009). "Errors in Operational Spreadsheets: A Review of the State of the Art," *42nd Hawaii International Conference on Systems Science* Kona, Hawaii.

Pressman, R. and D. Ince (2000). *Software Engineering: A Practitioners Approach* European Adaptation, McGraw Hill.

Pryor, L. (2003). "Correctness is not Enough," *EUSPRIG Building Better Spreadsheets from the Ad-Hoc to the Quality Engineered* Dublin, Ireland pp. 12-24, 1-86166-199-1.

Pryor, L. (2004). "When, Why and How to Test Spreadsheet Models," *Proceedings of EUSPRIG 2004 – Risk Reduction in End User Computing -'Best Practice for Spreadsheet Users in New Europe'* Klagenfurt, Austria pp. 24-35, 1-902724-94-1.

Rajalingham, K., D. Chadwick, B. Knight, and D. Edwards (2000). "A Spreadsheet Engineering Methodology," *Proceedings of the Thirty-Third Hawaii International Conference on System Sciences* Maui, Hawaii.

Reason, J. (1990). *Human Error* Cambridge University Press, Cambridge. UK, ISBN 0-521-3149-4.

Rust, A., B. Bishop, and K. McDaid (2006). "Investigating the Potential of Test-Driven Development for Spreadsheet Engineering," *Proceedings of EuSpRIG 2006 – Managing Spreadsheets: Improving Corporate Performance, Compliance and Governance* Greenwich, London pp. 95-107, ISBN 1-905617-08-9.

SERP (2006). *"Spreadsheet Engineering Research Project,"* Tuck Business School, Dartmouth College, http://mba.tuck.dartmouth.edu/spreadsheet/, USA.

Taylor, M., P. Moynihan, and T. Wood-Harper (1998). "End User Computing and Information Systems Methodologies," *Information Systems Journal* 8, pp. 85-96.

Thorne, S., M. Madahar, P. Cleary, D. Ball, C. Gosling, and K. Fernandez (2003). "Investigating the use of Software Agents to Reduce the Risk of Undetected Errors in Strategic Spreadsheet Application," *EUSPRIG 4th Annual Symposium* Dublin, Ireland pp. 147-159, ISBN 1-86166-199-1.

Thorne, S. and D. Ball (2009). "Example Driven Modelling for Decision Support Spreadsheets," *42nd Hawaii International Conference on Systems Science* Kona, Hawaii

Vemula, V. R., D. Ball, and S. Thorne (2006). "Towards a Spreadsheet Engineering," *Proceedings of EuSpRIG 2006 – Managing Spreadsheets: Improving Corporate Performance, Compliance and Governance* Greenwich, London pp. 95-107, ISBN 1-905617-08-9.

Yirsaw, A. and R. Mittermeir (2003). “Spreadsheet Debugging,” *Proceedings of EUSPRIG 2003 – Building Better Spreadsheets – ’From the Ad-Hoc to the Quality Engineered’* Dublin, Ireland pp. 71-85, 1-86166-199-1.

ABOUT THE AUTHORS

Dr. Simon Roy Thorne is a senior lecturer at the University of Wales Institute in Cardiff. Simon obtained his Ph.D. in July 2008 on the subject of alternative spreadsheet modelling techniques for the reduction of error in decision support spreadsheets. Simon is an active researcher in spreadsheet research with a particular focus of human factors in spreadsheets.

Dr. David Ball is a senior lecturer at the University of Wales Institute in Cardiff. David is a director of studies for several Ph.D. students in spreadsheet error research. David is an active in spreadsheet research with an emphasis on spreadsheet engineering.

Communications of the Association for Information Systems

ISSN: 1529-3181