A B C D E F G H I J K L

1 2 3 4 5 6 7 8 9 10 11 12 13

European Spreadsheet Risks Interest Group

Spreadsheet Risks
'the hidden corporate gamble'

Symposium:
18th - 19th July 2002
University of Wales Institute Cardiff, Wales, UK

UNIVERSITY OF WALES INSTITUTE, CARDIFF
UWIC
ATHROFA PRIFYSGOL CYMRU, CAERDYDD

Host:
University of Wales Institute Cardiff

Main Sponsor:
ISACA
(Northern UK Chapter)

Information Systems Audit and Control Association

SYMPOSIUM PROCEEDINGS EUSPRIG 2002

European Spreadsheet Risks Interest Group

EuSpRIG 2002 Symposium

"Spreadsheet Risks - the hidden corporate gamble"

Editor:
David Chadwick

Symposium Sponsored By

Symposium Hosted By

July 2002
Cardiff, Wales, UK

Proceedings of EuSpRIG 2002 Symposium
Spreadsheet Risks, Audit & Development Methods
First published 2002
Printed in Great Britain
Published by the University of Greenwich Press

Held at University of Wales Institute Cardiff, UK; July 18th-19th 2002

ISBN : 1 86166 182 7

Symposium Organised by
EuSpRIG Committee
in collaboration with
Information Systems Audit and Control Association (Northern UK Chapter)
www.isaca.org.uk/northern/

Cover design by David Chadwick at the University of Greenwich.

Proceedings of EuSpRIG 2002 Symposium
Spreadsheet Risks, Audit & Development Methods
First published 2002
Printed in Great Britain
Published by the University of Greenwich Press

Held at University of Wales Institute Cardiff, UK; July 18th-19th 2002

ISBN : 1 86166 182 7

Symposium Organised by
EuSpRIG Committee
in collaboration with
Information Systems Audit and Control Association (Northern UK Chapter)
www.isaca.org.uk/northern/

Cover design by David Chadwick at the University of Greenwich.

PREFACE

"Eusprig will never last beyond the first year" said a work colleague to me in July 2000 as he looked at a promotional pamphlet for our first symposium. 'Yes we will' I retorted and I am now so pleased to say Three years on 'I told you so'. But who would have believed it when we advertised our first symposium and waited nervously for people to show interest.

Perhaps one of the reasons we have kept going is that the main reason for our existence still exists. As Ray Butler, our first chairman pointed out in the preface to the first proceedings, 'Spreadsheet errors are still the rule, rather than the exception. Jobs and money are lost through business mistakes based on defective spreadsheet developments'. Ray's words ring as true today as in July 2000. We may not appear to be any closer to a solution to the general problem of errors but perhaps we, as EuSpRIG, are now bringing together more interested parties in the business and academic communities and starting to counter the perception of auditors that spreadsheets are not "proper systems", and by managers that spreadsheet developments are in some way trivial.

This year we have placed the emphasis on corporate risks because, in the end, this is where the spreadsheet errors have their effect – incorrect figures and interpretations leading to incorrect decisions affecting corporate welfare. And we have not been disappointed – this year we have received contributions from the UK, Austria, New Zealand and Canada. The message is being passed around the world and it is heartening to see people in both academia and industry responding and contributing to the general discussion. To this end, we hope that this years papers and management summaries will prove enjoyable and rewarding reading.

Yet again, we have received some excellent academic papers each of which has been reviewed by at least three referees with comments passed back to the authors. To encourage more input from professionals and practitioners we have this year encouraged the inclusion of management summaries in the proceedings. The message of the past two years has been that practitioners have plenty to say but not the time in which to write formal academic papers to be rigorously refereed; their preference was always to produce something shorter to be less rigorously vetted than formal papers. We have therefore this year included management summaries in the proceedings; we hope that this process will encourage more input from industry in the future.

Thanks are due to

- Patrick Cleary and Ray Butler who have carried the burden of organising this symposium,
- The members of the EuSpRIG committee for their support,
- Friends and supporters from our sponsoring bodies for their belief in us,
- All the session chairs.

Thanks also to all of you who have contributed papers and management summaries for inclusion in the timetable - without you there would probably be no EuSpRIG.

Please - enjoy the symposium, spread the word, share the knowledge.

David Chadwick, Chairman, EuSpRIG

June 2002

EuSpRIG Third Symposium on

Spreadsheet Risks - the hidden corporate gamble

ORGANISING COMMITTEE

Patrick Cleary, University of Wales Institute Cardiff, UK
Raymond Butler HM Customs and Excise

PROGRAMME COMMITTEE

David Chadwick University of Greenwich, UK
Len Arthur, University of Wales Institute Cardiff, UK
Grenville Croll, formerly of Andersen's, UK
Ray Panko, University of Hawaii, USA
Barry Pettifor, Price Waterhouse Coopers, UK
Ray Butler, HM Customs & Excise, UK

EUSPRIG MANAGEMENT COMMITTEE

David Ward, KPMG, UK
Graham MacDonald, HM Customs & Excise, UK
Grenville Croll, formerly of Andersen's, UK
Jocelyn Paine, Bristol University, UK
Leon Strous, De Nederlandsche Bank, Holland
Patrick O'Beirne, Systems Modelling Ltd, Ireland
Ray Butler, HM Customs & Excise, UK
David Chadwick University of Greenwich, UK

SPONSORS

Information Systems Audit and Control Association (Northern UK Chapter)
University of Wales Institute Cardiff, UK
Lloyds TSB
British Computer Society Information Risk Management & Audit Group

CHAIRMAN'S ARTICLE

A Future Role For EuSpRIG?

Papers in this section:

Training Gamble leads To Corporate Grumble ?

Quite simply, we do not think the way we think we think. Human cognition is built on complex mechanisms that inherently sacrifice some accuracy for speed. Speed in thinking was the overriding evolutionary force for our hunter ancestors, and although occasional errors caused the loss of hunters, new hunters were inexpensive and fun to make. Unfortunately, error rates acceptable in hunting and even in most computer applications cannot be tolerated in spreadsheets. For spreadsheet accuracy, we must somehow overcome or at least reduce human cognitive accuracy limitations dramatically.

Ray Panko: Spreadsheet Errors: What We Know, What We Think We Can Do

Training Gamble leads to Corporate Grumble?

David Chadwick, EuSpRIG Chair
D.R.Chadwick@gre.ac.uk

Fifteen years of research studies have concluded unanimously that spreadsheet errors are both common and non-trivial. Now we must seek ways to reduce spreadsheet errors. Several approaches have been suggested, some of which are promising and others, while appealing because they are easy to do, are not likely to be effective. To date, only one technique, cell-by-cell code inspection, has been demonstrated to be effective. We need to conduct further research to determine the degree to which other techniques can reduce spreadsheet errors.

As Ray Panko [1] correctly points out, the problem of spreadsheet errors has yet to be solved and represents a great, often unrecognised, risk to corporate decision making and financial integrity. Corporations are indeed gambling with the hidden risks. Abstracts of papers presented at past EuSpRIG symposia are available on the website www.eusprig.org ; a quick review of the titles suggests that the spreadsheet risk problem may be dealt with in several ways by:

- expanding awareness of the problem amongst corporations and academia Rajalingham et al [2], Ayalew et al [4], Butler R [6],Cleary P M [8]
- setting standards for end-user spreadsheet development Butler R [11] , Hawker A [3], Knight D [17]
- creating better audit approaches with accompanying software tools, Nixon D [18], Hock et al [7], Ettema H et al [16]
- Creating methodologies and software based tools for spreadsheet building Rajalingham et al [5] [14], Chadwick et al [10], O'Beirne [12], Paine [13], Raffensperger [15].

Many of the above are already being well researched and, no doubt, useful results will be obtained in due course but there is a view that the very least that can be done immediately is to concentrate effort in the first two areas mentioned. Expanding awareness and setting standards at an early enough stage in the training of would-be spreadsheet developers must surely have an effect in tackling the problem where it really begins - in human error. It is a belief held by the author that, often overlooked, is the need for better training and education of young professionals and students learning spreadsheet skills for the first time and surely destined to become the spreadsheet builders of tomorrow.

Many novice spreadsheet builders formally learn their skills by studying either for a professional examination of the auditing, IT and accountancy bodies or from courses provided at universities, colleges and private training agencies. If there is to be a true change in the quality of business spreadsheets then surely it is through such training and education providers that improvement and innovation must be channelled. The question, therefore, that needs to be addressed by EuSpRIG members is:

'How can EuSpRIG be actively involved in the teaching and assessment of spreadsheet skills where these occur?'

To help in answering this question it is useful to look at two areas where the influence of an interest group composed of active academics and professionals (such as EuSpRIG) may have an effect:

- the syllabi of professional examinations and coverage of spreadsheet risk,
- the curricula of higher education courses and coverage of spreadsheet risk.

1.0 Professional Examinations And Their Coverage Of Spreadsheet Risk

1.1 Professional Examinations

In the UK, there are several bodies either directly involved in the teaching and assessment of spreadsheet skills through their own professional studies or indirectly involved in the validating and influencing of syllabi of other educational providers. For instance, the ICAEW (Institute of Chartered Accountants England and Wales) is the main body influencing standards of assessment for accountants, the BCS (British Computer Society) is involved in setting its own examinations or in validating courses run at universities for IT professionals. So, too, the Institute of Internal Auditors (IIA) operates its own examinations for computer auditors. In terms of gaining a professional qualification in computer auditing and risk management there are probably two main avenues: the Certificate in Information Systems Auditing (CISA) and the Qualification in Computer Auditing (QiCA). Their approach to the development of skills differs. CISA appears suited primarily to those who have already gained some working knowledge by virtue of having done the practitioner job for some time. It results in a multiple-choice test which basically examines the applicant's in-depth knowledge of practical scenarios. It is therefore not suited to a beginner who has no or relatively limited personal experience. The QiCA, however, from the Institute of Internal Auditors, is comprised of two papers at different levels; the lower level paper is based upon learning from written texts and is more suited to the true novice, the practitioner with little experience or the student at university who both need to grasp fundamentals. The paper at the higher level deals with practical scenarios and is more appropriate for the experienced practitioner.

1.2 Training Curricula And Coverage Of Spreadsheet Risk

Neither the syllabus for CISA nor that for QiCA adequately cover the treatment of spreadsheet risks. A standard text used for teaching Computer auditing in the UK has one paragraph of four lines devoted to spreadsheet errors:

'Users of spreadsheets and database packages are necessarily immune from errors. 9%ile it is true that such packages provide a safe processing environment within which it is difficult if not impossible to make undetected or obscure input and output errors, it is still possible to make errors in logic. In fact, such errors maybe more difficult to detect than they would have been if a procedural programming language had been employed... This may be the case if a very large spread sheet is generated, so that only small parts of it can be viewed on screen at any one time... Chambers A.D and Court LM [9]

The third line admits that the problem for spreadsheets may be even worse than for normal software particularly with logical errors. Despite this, there is no literature on how to prevent errors nor on how to conduct a spreadsheet audit.

Perhaps this is where EuSpRIG has a role to play. Now established as the world’s foremost action group on spreadsheet risks isn't it about time that EuSpRIG set itself up as a standards making body, a pressure group to encourage the professional bodies to incorporate spreadsheet audit in their examinations

2.0 Higher Education Courses And Their Coverage Of Spreadsheet Risk

2.1 Universities Providing Courses Recognised By Professional Bodies

The professional bodies themselves provide distance learning courses or other arrangements for study but there are several universities around the UK who offer courses preparing students either to sit for the professional audit examinations directly or to gain exemption from the same by passing other recognised courses. City Business School, Guild Hall University, University of Central England and Sheffield Hallam University are but a few of those that provide courses preparing candidates to sit the two papers of the IIA's Qualification in Computer Auditing.

Other UK universities, such as Southampton Institute and University of Greenwich, take a different approach. They have applied to the IIA for recognition of their existing undergraduate computing courses covering auditing material at the lower level of the QiCA syllabus and requesting that these courses be recognised as equivalent to the IIA's own professional exam. At the University of Greenwich the HA has permitted students passing two particular undergraduate courses to apply for exemption from HA studies at the lower level, an arrangement has been extremely beneficial for the teaching of computer auditing at the university.

2.2 Spreadsheet Teaching At Universities

There is no doubt that spreadsheet teaching at universities suffers from an image problem:

- spreadsheets are seen as trivial encompassing simple accounting models,
- the packages are easy to learn therefore the intellectual content is limited,
- students learn basic skills prior to university and have developed bad habits.

But perhaps the major problem is that of convincing academics that spreadsheet teaching requires a complete re-think involving not only changes in teaching content but also changes in teaching method perhaps accompanied by the development of a research ethos to find appropriate and innovative solutions.

2.3 Encouraging A Research Ethos

Possible benefits of developing a research ethos are shown below (Fig.1).

Fig. 1 Rationale For Developing A Research Ethos

The rationale for developing a spreadsheet research ethos is that it encourages a Research-Teaching Feedback Loop (Fig.2). Immediate improvements occur in feedback from research to classroom and feedback from classroom to research with the overall effect being an improvement in both research and teaching leading to an improvement in student learning.

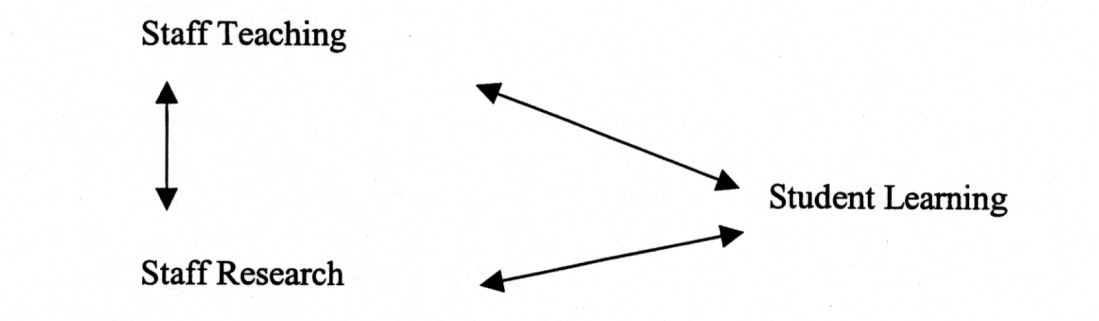


Fig 2: Research-Teaching Feedback Loop

The above approach necessitates a close liaison between teaching and research which in many universities tend to operate independently of each other. What is required is an approach whereby the teachers and the researchers work together to improve both the taught content (i.e. syllabus, the actual substance of imparted knowledge) and also the teaching method (i.e. types of materials, teaching style, organization of the student cohort).

2.4 Integrating Research And Teaching

Recent research has highlighted the high incidence of errors in spreadsheet models used in industry. In an attempt to reduce the incidence of such errors, a teaching approach has been devised which aids students to reduce their likelihood of making common errors during development. The approach comprises of spreadsheet checking methods based on the commonly accepted educational paradigms of peer assessment and self - assessment. However, these paradigms are here based upon practical techniques commonly used by the internal audit junction such as peer audit and control and risk self - assessment. The result of this symbiosis between educational assessment and professional audit is a method that educates students in a set of structured, transferable skills for spreadsheet error checking which are useful for increasing error-awareness in the classroom and for reducing business risk in the workplace

Chadwick D, Sue R [10]

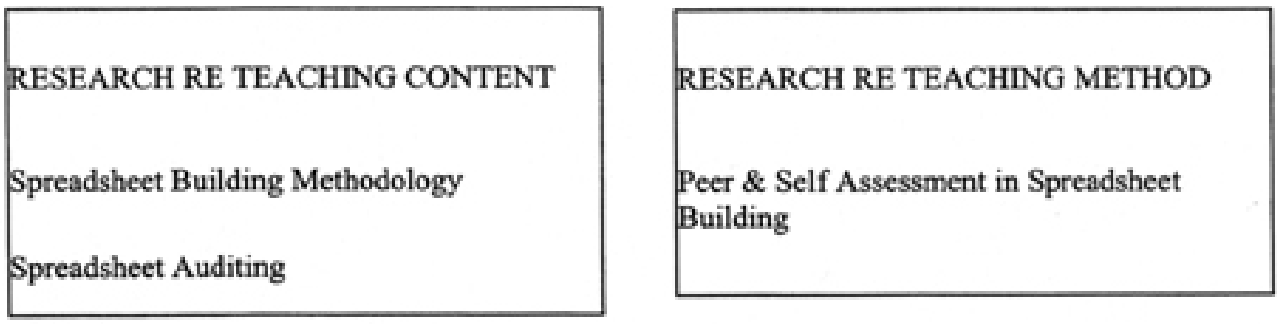


Fig 3. Two Research Strands

The University of Greenwich has developed its own model (Fig.3) on integrating teaching content and teaching method and also on integrating educational research with formal software engineering research with regard to spreadsheet risks (Fig. 4). In the latter there is classroom observation of students and the errors they produce whilst building spreadsheets. These findings are fed into the formal software engineering research process which is currently working on a software engineering methodology for developing spreadsheets as pieces of software. As the formal research progresses so it spins off new classroom techniques which are tried and tested and feedback given to the formal research. The classroom experience itself produces valuable findings for formal research within the arena of education.

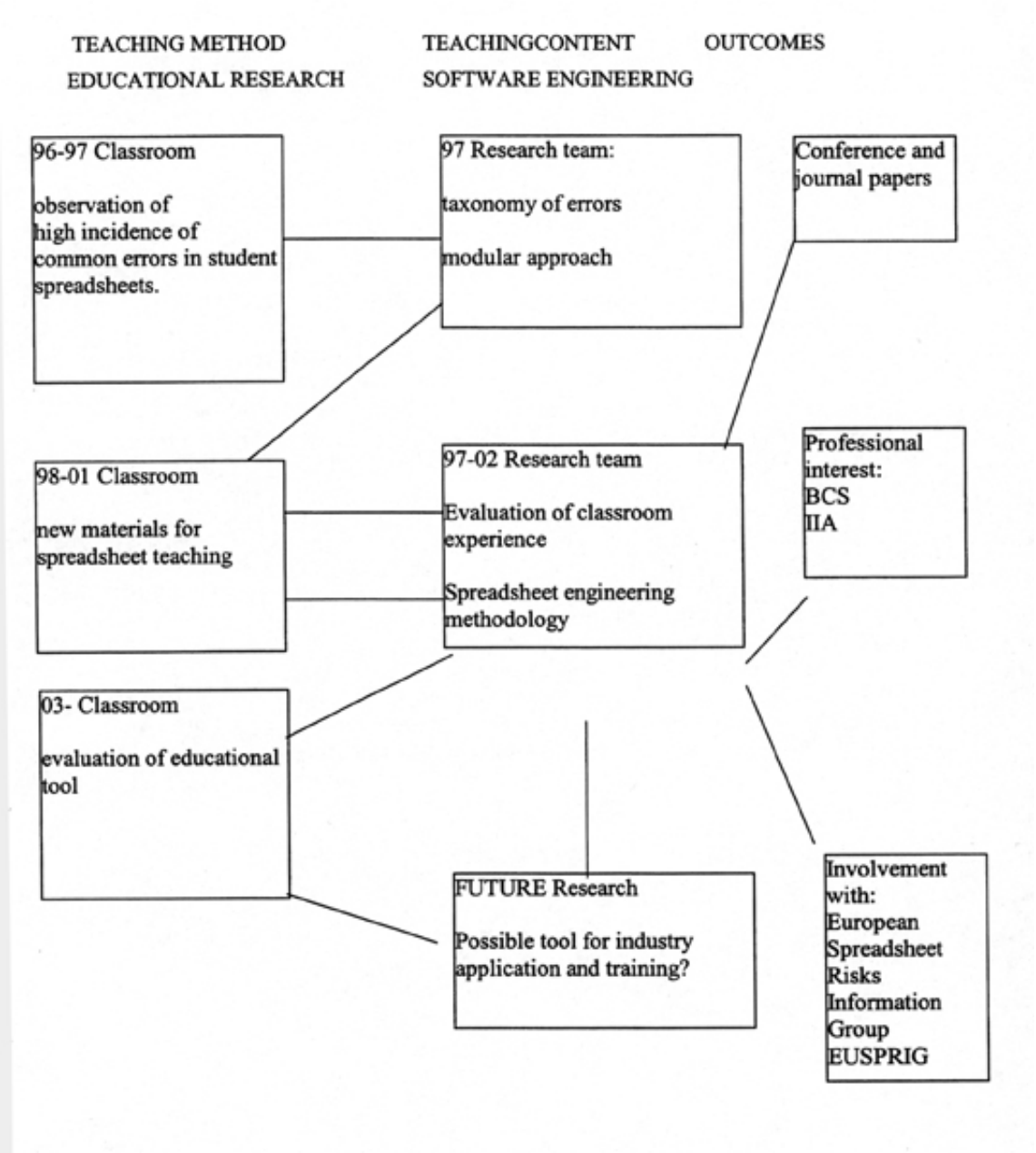


Fig 4: The University of Greenwich model showing integration of educational and software engineering research

3.0 The Importance of Standards In Training Regimes

So far we have addressed the methods of training offered by professional bodies and educational institutions. It is necessary now to look at what needs to be taught and the question of providing standards. Standards concerning spreadsheet development and audit are very important because without them, nobody would be clear just exactly what needs to be taught and how it is to be assessed. This topic generally needs more thought and debate than was possible in the preparation of this paper but suffice it to say that the matter has been brought to the attention of EuSpRIG delegates by Ray Butler at the Amsterdam symposium.

One of the problems reported by researchers and auditors in the field of spreadsheet risks is that of getting and keeping management's attention to the problem. Since 1996, the Information Systems Audit & Control Foundation and the IT Governance Institute have published CobiT ® which brings mainstream IT control issues into the corporate governance arena. This paper illustrates how spreadsheet risk and control issues can be mapped onto the CobiT framework and thus brought to managers' attention in a familiar format. Butler R [11]

There is a need for the widespread promotion and eventual adoption of sound spreadsheet building and spreadsheet auditing standards. From a look at the training and teaching curricula of academia and higher education, there appears to be no attempt to present students with working standards for spreadsheet developments nor even an acknowledgement that such may be required. It seems pertinent therefore that EuSpRIG should approach these bodies and suggest that firstly, more emphasis be placed upon spreadsheet errors and that secondly, good practice techniques be promoted wherever possible. In due course perhaps EuSpRIG itself could suggest tried and tested guidelines of its own. The CobiT framework seems a useful starting point for EuSpRIG endeavours in this area.

4.0 How can EuSpRIG Be Involved In Education and Training?

Finally, let us consider where EuSpRIG can go from here. As an interest group composed of professional and academic contacts and with worldwide affiliations it may be able to play a decisive role in encouraging educational agencies to change their approach to spreadsheet skills training, education and assessment. At a minimum and with little effort initially EuSpRIG could be become more involved with:

- promoting a greater emphasis to be placed upon spreadsheet teaching,
- promoting research through encouragement of participation in EuSpRIG ,
- using its influence to seek funds for research-active EuSpRIG members.

5.0 Conclusions

The crux of this paper has been to look at perhaps the most straightforward and immediate ways in which EuSpRIG can grow and contribute directly to the problem of spreadsheet risks.

EuSpRIG has a role to play:

- in the development of syllabi in professional studies,
- in curriculum development and research in higher education,
- in the creation and promotion of spreadsheet building and auditing standards.

6.0 References

[1] *Spreadsheet Errors: What We Know. What We Think We Can Do.*

Panko R Proceedings of 1st International Symposium on Spreadsheet Risks, EUSPRIG2000, Greenwich, UK, July 2000. http://arxiv.org/abs/0802.3457

[2] *Classification of Spreadsheet Errors*

Rajalingham K, Chadwick D, Knight B: Proceedings of 1st International Symposium on Spreadsheet Risks, EUSPRIG2000, Greenwich, UK, July 2000. http://arxiv.org/abs/0805.4224

[3] *Building Financial Accuracy into Spreadsheets*

Hawker A: Proceedings of 1st International Symposium on Spreadsheet Risks, EUSPRIG2000, Greenwich, UK, July 2000. http://arxiv.org/abs/0805.4219

[4] *Detecting Errors in Spreadsheets*

Ayalew Y, Clermont M, Mittermeir R: Proceedings of 1st International Symposium on Spreadsheet Risks, EUSPRIG2000, Greenwich, UK, July 2000. http://arxiv.org/abs/0805.1740

[5] A Structured Methodology for Spreadsheet Modelling.

Rajalingham K, Chadwick D, Knight B; Proceedings of 1st International Symposium on Spreadsheet Risks, EUSPRIG2000, Greenwich, UK, July 2000. http://arxiv.org/abs/0805.4218

[6] Risk Assessment for Spreadsheet Developments

Butler R (HM Customs and Excise) Proceedings of 1st International Symposium on Spreadsheet Risks, EUSPRIG2000, Greenwich, UK, July 2000. http://arxiv.org/abs/0805.4236

[7] *Visual Checking Of Spreadsheets*

Chan H Q Chen Y: Proceedings of 1st International Symposium on Spreadsheet Risks, EUSPRIG2000, Greenwich, UK, July 2000. http://arxiv.org/abs/0805.2189

[8] *How Important Are Spreadsheets To Organisations?*

Cleary P M: Proceedings of 1st International Symposium on Spreadsheet Risks, EUSPRIG2000, Greenwich, UK, July 2000

[9] *Computer Auditing*

Chambers A.D and Court LM *(1994)* Computer Auditing 3rd Edition. Pitman.

[10] *Teaching Spreadsheet Development Using Peer Audit and Self-Audit Methods for Reducing Errors*

Chadwick D., Sue R.: Proceedings of 2nd International Symposium on Spreadsheet Risks, EUSPRIG2001, Amsterdam, Holland, July 2001. http://arxiv.org/abs/0801.1514

[11] *Applying the CobiT ® Control Framework to Spreadsheet Developments*

Butler R J : Proceedings of 2nd International Symposium on Spreadsheet Risks, EUSPRIG2001, Amsterdam, Holland, July 2001. http://arxiv.org/abs/0801.0609

[12] *Euro Conversion in Spreadsheets*
O'Beirne P: Proceedings of 2nd International Symposium on Spreadsheet Risks, EUSPRIG2001, Amsterdam, Holland, July 2001

[13] *Safer Spreadsheets with Model Master*

Paine J: Proceedings of 2nd International Symposium on Spreadsheet Risks, EUSPRIG2001, Amsterdam, Holland, July 2001. http://arxiv.org/abs/0801.3690

[14] *An Evaluation of the Quality of a Structured Spreadsheet Development Methodology*
Rajalingham K, Chadwick D, Knight B: Proceedings of 2nd International Symposium on Spreadsheet Risks, EUSPRIG2001, Amsterdam, Holland, July 2001.
http://arxiv.org/abs/0801.1516

[15] *New Guidelines for writing spreadsheets*
Raffensperger J: Proceedings of 2nd International Symposium on Spreadsheet Risks, EUSPRIG2001, Amsterdam, Holland, July 2001

[16] *Assurance by Control around Is a Visible Alternative to the Traditional Approach*
Ettema H, Janssen P, de Swart J: Proceedings of 2nd International Symposium on Spreadsheet Risks, EUSPRIG2001, Amsterdam, Holland, July 2001. http://arxiv.org/abs/0801.4775

[17] *A Real alternative to spreadsheets*
Knight D,: Proceedings of 2nd International Symposium on Spreadsheet Risks, EUSPRIG2001, Amsterdam, Holland, July 2001

[18] *Spreadsheet Auditing*
Nixon D, O'Hara M: Proceedings of 2nd International Symposium on Spreadsheet Risks, EUSPRIG2001, Amsterdam, Holland, July 2001

ACADEMIC PAPERS

All papers in this section have passed a rigorous review process. Each has been closely reviewed by at least three referees.

Papers included are those deemed to provide a significant contribution to understanding of the problem of spreadsheet risks and their solution.

Interpretation As A Factor In Understanding Flawed Spreadsheets
David Banks, Ann Monday
University of South Australia, Adelaide, Australia

Spreadsheet Engineering : A Research Framework
Thomas Grossman
University of Calgary, Alberta, Canada

A Spreadsheet Auditing Tool Evaluated In An Industrial Context
Markus Clermont, Christian Hanin, Roland Mittermeir
University of Klagenfurt, Austria

Spreadsheets are often hard to read and spreadsheets often contain mistakes. Panko and Sprague, 1998, found that around 2% of spreadsheet cells contain errors. The likelihood of a correct model is less than 50% for even a spreadsheet with only 35 cells. Why are spreadsheets so difficult?
John F. Raffensperger, New Guidelines For Spreadsheets, EuSpRIG 2001

Interpretation as a factor in understanding flawed spreadsheets

David A. Banks and Ann Monday
University of South Australia, Adelaide, Australia
Email {David.Banks}, {Ann.Monday}@unisa.edu.au

ABSTRACT

The spreadsheet has been used by the business community for many years and yet still raises a number of significant concerns. As educators our concern is to try to develop the students skills in both the development of spreadsheets and in taking a critical view of their potential defects. In this paper we consider both the problems of mechanical production and the problems of translation of problem to spreadsheet representation.

1. INTRODUCTION

The spreadsheet has been used as a decision support tool for a considerable number of years but still suffers from significant defects in terms of the way that they 'inform', or arguably 'misinform', decision makers. Spreadsheets, at least in the hands of the typical end user, have proved to be less effective as decision support tools than may have been expected. Although they apparently offer decision makers the ability to develop models of business problems and to test various possibilities by manipulating the data in the model, two key problem areas emerge in practice. These areas relate to errors made in the mechanical production of the spreadsheet and also to errors in the translation of the problem situation into a model that is then represented as a spreadsheet. These two causes of error can lead to a position where decision makers may act in the belief that decisions can be made with confidence on the output from the spreadsheet despite evidence to the contrary. In this paper we consider both the problems of mechanical production and the problems of translation of problem to spreadsheet representation, with a strong interpretive focus on the latter. Our approach is therefore biased towards the 'softer', interpretive dimensions of the information systems field and views the spreadsheet as being a tool located in a socio-technical system. We do, however, also draw upon some of the 'hard' programming literature as this provides useful insight to some aspects of the spreadsheet development process. We develop our considerations primarily within the context of the teaching of subjects that utilise spreadsheets as decision support tools and illustrate our arguments with examples of student work.

2. PROBLEMS IN SPREADSHEET DEVELOPMENT

Since the proliferation of PCs in the early 80s businesses have seen a continuing growth in the use of applications created by end-users for themselves and others in their organisations. Frenzel (1992) argues that User Developed Applications (UDAs) have grown because they offer greater user control and increased flexibility and responsiveness for the user. They encourage innovation, reduce the workload of the IT department and enable the user to achieve what they want in a much shorter time-scale. Thus they allow changes to be implemented quickly and the effects to be seen immediately. McGill (2000) notes that UDAs now form a significant proportion of organisational information systems. The downside of this faster and user directed development has, however, led to

the emergence of major problems that can have significant and negative impact on businesses that rely heavily on spreadsheets.

Alavi and Weiss (1985) note that 'In the enthusiasm to benefit from EUC activities, corporations are overlooking the potential risks of these activities. Organizational exposure to EUC is costly.' They point to problems of lack of time devoted to problem definition, lack of specification, solving the wrong problem, selection of inappropriate software tool, lack of documentation, lack of testing and validation etc. The effect of poor development, testing and maintenance processes can typically impact upon bottom line values, tax payments, negative balance for stock on hand and significant underestimates in project costings (Panko and Halverson, 1996). Panko (2000) cites a variety of studies that indicate a high percentage of all spreadsheets contain errors. Audit work carried out by consulting organisations have found that up to 90% of all spreadsheets with more than 150 rows contained errors. Many of these spreadsheets were of high importance to the organisations concerned. One survey of 106 spreadsheets indicated that 49% of the surveyed spreadsheets created new corporate data and only 7% of spreadsheets were considered to be of 'low' importance (Hall, M.J.J. 1996, cited in Panko and Halverson 1996).

Other research examined the risks associated with spreadsheet errors. Teo and Tan (1999) found that '…spreadsheet applications were used by an overwhelming 91% of a sample of end-users', whilst Janvrin and Morrison (2000) comment that '...as many as 44% of all end-user spreadsheets contain at least one error'. Since 1987 researchers have shown that the problem of spreadsheet errors in UDAs is increasing. Teo and Tan (1999) state that 'This growing concern over spreadsheet errors can be attributed to the increasing popularity of spreadsheets to support important financial analysis, budgeting, and forecasting applications.' They have also seen '...a tendency for end-users to view spreadsheets as simple tools and to be overconfident about the error-free nature of their spreadsheets.' McGill (2000) noted UDAs that may be incorrect in design, inadequately tested, and poorly maintained. Teo and Tan (1999) also found that 'inadequate care is taken to design spreadsheets, which in turn makes error detection and correction even more difficult' and the quality of the application is reduced further.

Edberg and Bowman (1996) recognised that 'UDAs represent a considerable risk to organizations since users who create applications frequently have little or no training in development methods'. The need for end-users to be given some design and implementation training is highlighted by Hobby (1996). Ross and Ruhleder (1993), in exploring the range of skills required by IS professionals suggest that IS educators should impart not only the technical skills to students but also provide them with an awareness of a wide range of technical, social and organisational concerns.

Further research into spreadsheets has included Chan and Storey (1996) who investigated the use of spreadsheets within the organisation to determine the relationship between usage, tasks, proficiency and satisfaction. This research determined 'proficiency has a greater impact on the tasks than the tasks have on proficiency; users do not often use the commonly available spreadsheet features; users prefer software packages they know and understand rather than the best package for the task; user proficiency was not related to the importance of the decision made as a result of the spreadsheet analysis'.

There are a number of techniques and tools that can be used to verify at least the mathematical or formulaic integrity of spreadsheets, based on their quantitative nature. The causal relationships of the data in spreadsheets can be rigorously investigated and tests can be applied to check that with given input data the process modelled in the

spreadsheet will provide an accurate output. It is thus possible to check that the data is correct and is in the required format, that the relationship of the data is logical, formulae are valid and so on. These can be viewed as representing the 'hard' aspects of the spreadsheet and errors here may be attributed to typing errors, incorrect linking of cells, incorrect use of formulae and a variety of other technical causes. However, our work with student spreadsheets indicates that while these hard errors do occur the main problem lies in the translation, or interpretation, of a given problem into a spreadsheet representation of that problem. Our purpose in using spreadsheets is to develop their understanding of a spreadsheet as not just a quantitative, number processing tool, but also as a qualitative tool that informs the decision maker rather than providing the answer.

Our experience with students over a number of years shows they continue to make such common errors in spreadsheets as identified in taxonomies of errors (Rajalingham, Chadwick and Knight 2000 and Panko 2000) - including poor cell protection (often none at all), poor use of formulae (including embedded numbers), lookup poorly used, no validation, clear inconsistencies in data output that should have been checked and assumptions not clearly stated.

We also consistently see attempts to compare 'apples with oranges' and incorrect conversions of data into common format (litres/gallons, kilograms/tonnes, dollars/yen). Had these students been developing the spreadsheet in a real business setting it is evident that the results would have had little real value for a decision maker. They made an assumption that there was a single 'right' answer, and that their model generated this 'right' answer. This was despite the fact that they were told the spreadsheet would be used as a decision *support* aid for a group of managers and would, in all probability, not provide a single definitive answer. There is a tendency to adopt a view of spreadsheet development that echoes Rushby's (1980) comments about computer program development: 'It is tempting, after the white heat of coding, to assume that if the program compiles and runs without any execution errors, then it works. Perhaps it does work, but all that proves is that the program does something – not that it does the right thing. Only thorough testing can we evaluate the program, and demonstrate that it does indeed meet its' specification'

We are not suggesting that students are incapable of developing good spreadsheets. Our courses are not training courses and seek to expose students to the difficulties of developing decision support models using a tool that itself has a number of limitations. Klein and Methlie (1995) suggest that the problems with spreadsheets in the context of decision support systems include:

- "they do not provide a clear understanding of an application global logic in using resources such as data, decision models, reports, forms, etc;
- they do not provide satisfactory readability of decision models;
- they do not provide a way to represent easily data structures more complex than two-dimensional tables;
- they prevent easy evolution of the application due to the non-separation between data management models, form and report definition. All these tasks must be accomplished by the user within a grid of cells, a cell being able to contain a piece of data, a formula and be used for presentation."

They conclude that the consequence of these limitations is that spreadsheets may be useful for simple personal applications but are very risky for more complex or institutionalised applications. It is these complex problems that exist in a socio-political

business setting that are of interest to us. In the next section we introduce interpretation as a significant factor in the complex decision environment of modern business.

3. INTERPRETATION AS A COMPLICATING FACTOR

An individual can be seen as having a socially constructed, or nominalist, view of the world (Hirschheim and Klein 1989) and it is this social construction with its particular set of 'values, outlook, the worldview, (Weltanschauung), which makes a particular model meaningful' (Checkland and Holwell 1988). Without an appreciation of the worldview that the spreadsheet models were generated within it is unlikely that the full tacit connections will be obtained from the explicit content of the spreadsheet.

The worldview of the spreadsheet developer may also have been intrinsically flawed or may have been incorrectly articulated, captured or recorded. An individual 'expert' taking a defined role within an organisation may be acting with a distorted view of the world, but if there are no discernable dissonances then 'the performer can be fully taken in by his own act; he can be sincerely convinced that the impression of reality which he stages is the real reality' (Goffman 1978). If it is a respected role within the organisation, for example that of an 'expert', or experienced or senior member of the organisation then Goffman suggests that 'the audience is also convinced in this way about the show he puts on … then for the moment at least, only the sociologist or the socially disgruntled will have any doubts about the 'realness' of what is presented'. If the 'expert' is responsible for recording their knowledge into the spreadsheet then there is also the problem of actors presenting to the system only the polished and packaged end product of their reasoning. The errors and mistakes made during the reasoning process will have been corrected such that the 'telltale signs that errors have been made and corrected are themselves concealed … In this way an impression of infallibility, so important in many presentations, is maintained' (Goffman 1978)

If knowledge, then, is taken as 'dynamic human processing justifying personal belief toward the "truth" (Takeuchi and Nonaka in (Morey et al 2000)) developers must view the output of a spreadsheet as 'justified personal belief' rather than 'truth'. If an individual believes something to be true and acts as if it were true and as a result of that action perceives confirmatory outcomes, then they will continue to believe their perception to be real. Even if the spreadsheet output is incorrect there may be political or social reasons to project the outcome as if it were correct and this aspect of the process needs to be recognised as a contributory factor in the over analysis of the failure of spreadsheets. Kerzner (1998) notes that one implementation problem experienced in reporting project status or progress is that 'upper level personnel generally prefer the more traditional methods [of non-electronic reporting supported by trial and error], or simply refuse to look at reality because of politics'. He suggests that this political agenda may lead to data submitted to the board being based upon 'an eye-pleasing approach for quick acceptance, rather than reality'.

Panko and Halverson (1996) note that 'developers may even include deliberately incorrect data, or at least data from estimates that are dubious but support their cases' and that there is a natural tendency to select assumptions that fit individual expectations and desires. This indicates that even with a quantitative decision support tool the users are engaging in qualitative or interpretive actions and that these actions can significantly distort decision processes that are based on these actions and potentially lead to poor, ineffective or possibly dangerous business decision-making.

4. HELPING STUDENTS DEAL WITH COMPLEX AND INTERPRETIVE DEVELOPMENT

The operation of software tools is quite clearly a training specific activity, whereas the business problem solving approach would seem to fall more comfortably into the domain of education in the broader sense. Business computing has been in the curriculum of higher education for some considerable time, so it is rather surprising that the results of this exposure are not as successful as would be hoped for. Students do need software-specific skills but these must be matched with an educational process that helps them to contextualise those skills within a socio-technical business setting that is characterised by complexity, change and political action.

The difficulty that typically faces higher education is the need to set problems that are pre-digested and have predictable outcomes, ie a 'correct' answer against which a students performance can be measured. There is also often no effective way to simulate true business problems whose features often include perceived complexity, ambiguity, time and political pressure and so on. Such aspects of the real world are difficult to model within educational systems which are bounded by time and assessment implications, but our own work in this area suggests that considerable benefits can accrue if students are given problems to which there is no single correct outcome and where the exercise alerts them to the fact that in some cases a spreadsheet can only offer a guide rather than an accurate answer. Practical difficulties of assessment do arise, as the focus of activity becomes more process than product centred, and this becomes more difficult to measure in an objective fashion, but we argue that the benefits should outweigh these considerations. Greater concentration on the development of problem solving skills rather than offering a route to 'quick and dirty' implementation of an obvious answer to a problem should help students to become more careful in their adoption of simplistic, and potentially flawed, models, which are later implemented on a computer system.

4.1 Examples of the problems used and results obtained

Balls in boxes – overly simplified or impossibly complex attempts to solve

The basic problem here was based around the idea of packing balls in boxes, eg how many balls of a specific diameter could be packed in a box of given dimensions. Variations of this problem used mixtures of boxes and drums for packing and were presented as an industrial pellet production and packing process. Solutions developed ranged from simple volume division (volume of box divided by volume of ball multiplied by number of balls) through to attempts that recognised complex matrix packing variations for different sizes of ball. Perhaps the most creative solution was one that produced, for one specific combination, an answer that identified the optimum packing container as a drum, but pointed out that from a wider shipping perspective boxes would be more efficient. Some students were embarrassed to admit that they used marbles and jam-jars filled with water to explore the problem – they felt that this was quite a childish approach even though we praised them for their sensible experimentation. (Local stores did complain that they were selling out of marbles, even though this game was ‘out of season’!) The solutions thus represented a spectrum from overly simplistic (the volume division approach) through to one that was too complex to model (the matrix model). Students found it difficult to accept that there was no ‘perfect’ answer to the question, tending to believe that what was required was further refinement of their solution rather than re-examination of the problem itself. These outcomes echoed previous work carried out by one of the authors with adult learners who were learning to develop computer programs in BASIC (Banks, 1988).

Machine selection – interpretation of data

We have also used a complex 'machine selection' problem in which students were required to develop, as groups, a spreadsheet to support a group of decision makers who were selecting one of five industrial production machines. With the same given information and specifications, of 100 groups of students 38 opted for machine A, 14 for machine B, and sixteen each for the remaining three machines. The spreadsheets contained few technical errors, in terms of errors that would cause the spreadsheet to malfunction, with the major differences in outcomes being a result of the interpretations that the students had applied to the data provided. The range of problems that were evident in the solutions for this problem included many of those mentioned in broader spreadsheet literature, including: poor design and layout, overly complex solutions that would be difficult for a group of users to understand, not comparing like with like, missing items in comparisons, assumed a 'right' outcome, little on-screen guidance for users, input and output data too far apart, little use of graphing, too much data on screen, incorrect filename protocol, ranking and weighting applied, and a good concept, but poorly executed.

4.2 The student approach

In addition to the mechanical problems raised earlier the following issues have been noted:

- that students tend to develop the outlines quickly without any real planning or problem solving.
- they are unable to filter out the data that is not required to solve the problem, attempting to incorporate all data provided. In particular, students are reluctant to remove data that has been provided for them.
    - students do not appear to recognise the importance of the interpretive act in both the design and the assessment of the validity of the output, believing their solution to be correct.
- students spend more time on presentation than testing the output, and apply fashion colours rather than good working practices.

However, in terms of attitude to this type of problem solving, many students who have been reticent in the early stages of the project, have risen to the challenge and developed considerably in terms of confidence in tackling unfamiliar and more complex problems.

4.3 Problems of teaching

In attempting to expose students to real world, complex problems we introduce risk, not only for the students, but for ourselves and our teaching team. It has not been uncommon for some support teaching staff to seek 'the answer' to the problem and to find it difficult to see that there are a number of answers based on the interpretive approach adopted by the students. We suspect that some tutors have found it embarrassing to admit that they do not know 'the' answer and have therefore tended to provide students with their own interpretations of the situation, these being accepted by the students as 'true' solutions. We have found it relatively easy to design simple spreadsheet problems but much more difficult to develop complex problems that stretch the students but can also be contained within a relatively short teaching semester.

4.4 How can we improve this situation?

Our aim is to help students, who will enter the business world as both constructors of spreadsheets and as consumers of the output from other developers' spreadsheets, to acquire a view of spreadsheets that acknowledges their value but also recognises the potential problems. For students who will become consumers of spreadsheets an option would be to provide them with constructed spreadsheets that produce different outputs for the same problem and ask them to evaluate the different scenarios.

For constructors of spreadsheets we must consider alternative approaches to developing skills in the design and the mechanical development of spreadsheets. We must also explore ways to emphasise the importance of documentation, testing and auditing of spreadsheets, particularly larger models; recognition that the models built are a product of the developers worldview rather than a 'scientific' or 'rational' process; recognition that the data present in spreadsheet and the way it is processed may not truly reflect the 'real' data as a result of deliberate or accidental misrepresentation of that data; and recognition that the data produced from spreadsheets is interpreted within the context of a broader socio-political business environment where even accurate output may be deliberately distorted.

This is a difficult task to embark upon but all of these factors must be taken into account when trying to understand why spreadsheets are causing such concern and potential damage in the business community. Computer programming has already been through much of this reflective thinking and spreadsheet developers, who carry out similar tasks to programmers in the development of a spreadsheet model, can learn from their conclusions. Dick (1985) offers a useful parallel in the need for a change of educational emphasis in his discussion of education for engineers: 'While engineering education has done a good job of teaching engineers how to use the tools of analysis, it has provided little or no training in the mechanics of forming and manipulating concepts. This has led many engineers to settle for mediocre results, rather than strive for more elegant, path-breaking solutions.' He suggests that the difference between inspired and mundane performance is not academic, it is more the application of both detailed analysis and conceptualisation to a problem to generate a creative solution and he states that: 'An engineer who consciously applies conceptualisation skills is better at defining and assessing problems, developing alternative solutions, and determining the best solutions. These abilities, in turn, are the basis for such breakthroughs as using old products in new ways or applying new technologies to revolutionize existing designs'

Another critic of the rigid thinking patterns produced by some engineering training and education is Sargent (1990) who comments that: 'Conventional engineering sciences and mathematics are oriented to understanding the physical world in which engineering occurs, and at applying that knowledge in techniques of analysis. Also many mathematical representations of the world are taught in which the "best" design is implicit, and can be determined by solving the relevant equations. Sargent (1990) continues: 'However, where the essence of the problem does not have a convenient representation, then degree courses are very poor in teaching methods of synthesis, of generating good potential designs which can then be analysed by more formal methods.' He argues that, in the real world, proposed designs '...will involve a variety of problems which cannot all be represented in the same mathematical description, and which may involve currently intractable problems which can be solved only approximately.' For such circumstances he indicates that the systematic generation of design alternatives and the selection of better ones based on incomplete information offers a productive path. The application of such approaches by end users in the

computing world should help avoid some of the basic design problems that clearly plague even such apparently simple applications as spreadsheets.

5. CONCLUSION

Spreadsheets appear to be a relatively intuitive tool that are perceived to be deceptively easy to use and although extensively used by organisations have often proved to be misleading at best and dangerous at worst. The spreadsheet itself has some limitations in its ability to act as a vehicle for the development of complex business models and users may be relatively naï ve in the development of models. Their basis for developing the model may be based on a unique worldview, or contain deliberate or unintended biases. The model may have been deliberately tampered with to generate acceptable figures for a specific instance to preserve a perceived 'expert' label for the developer.

Unless we can help students recognise these broader interpretive components of the process as well as use good mechanical design principles the problems of poorly developed and applied spreadsheets will not be alleviated in the near future. Spreadsheets are clearly useful to business but unquestioning reliance on their output is clearly dangerous given the possible obstacles that lie in the way of the development of a reliable output. We need to help future developers appreciate all of the literal and interpretive problems, social, political and technical, that influence the development of spreadsheets and to avoid the trap of developing applications that contain disasters waiting to happen. Above all we must avoid the mentality of the developer who left the comments below embedded in a large database application that ran successfully for one year after his efforts but then failed totally:

*apologies to anyone trying to maintain this code
*I just coded it and debugged it by running, and then
*added further pieces into it as they were needed.
*Anyway, I reckon that the more confusing it looks
*the better it must be!!!!!!!!!!

6. REFERENCES

Alavi, M. and Weiss, I.R. (1985), "Managing the Risks Associated with End-User Computing", Journal of management Information Systems, Vol II, No 3

Banks, D.A. (1988), "Problem Solving in the Teaching of Computing", Computers in Adult Education and Training, 49-53

Chan, Y.E. and Storey, V.C. (1996), "The use of spreadsheets in organisations: Determinants and Consequences", Information and Management, 31, 119-134

Checkland, P.B. and Holwell, S. (1998), Information, Systems and Information Systems: Making Sense of the Field, Wiley, Chichester, UK

Dick, M. (1985), "Creative problem-solving for engineers", Machine Design, Feb 7

Edberg, D.T. and Bowman, B.J. (1996), "User-Developed Applications: An Empirical Study of Application Quality and Developer Productivity", Journal of Management Information Systems, 13,1, 167-185

Frenzel, C.W. (1992), Management of Information Technology, Boyd & Fraser, 294

Goffman, E. (1978), The Presentation of Self in Everyday Life, Pelican Books, Middlesex, UK

Hirschheim, R. and Klein, H.K. (1989), Communications of the ACM, 32, 10

Hobby, J (1996), "Degrees of Excellence", Computer Weekly, 02/08/1996, 32

Janvrin, D. and Morrison, J. (2000), "Using a structured design approach to reduce risks in end user spreadsheet development", Information and Management, 37, 1-12

Kerzner, H. (1998), Project Management: A Systems Approach to Planning, Scheduling and Controlling, Wiley, NY

Klein, M.R. and Methlie, L.B, (1995), Knowledge-based Decision Support Systems, Wiley**,** 185

McGill, T. (2000), "User Developed Applications: Can End Users Assess Quality", Proceedings of the 2000 IRMA International Conference: IT Management in the 21st Century, Alaska, 106-111

Morey, D., Maybury, M. and Thuraisingham, B. (Eds.) (2000) Knowledge Management: Classic and Contemporary Works, MIT Press

Panko, R.R. (2000), "What We Know About Spreadsheet Errors", [accessed on line http://panko.cba.hawaii,edu/ssr/Mypapers/whatknow.htm, 3.39 pm 5/2/02]

Panko, R.R. and Halverson, R.P. Jr. (1996), "Spreadsheets on Trial: A Survey of Research on Spreadsheet Risks", Proceedings of the Twenty-Ninth Hawaii International Conference on System Sciences, Maui, Hawaii, 326-335

Rajalingham, K., Chadwick, D.R. and Knight, B. (2001), "Classification of Spreadsheet Errors", Proceedings of the EuSpRIG 2001 [accessed on line http://www.gre.ac.uk/~cd02/EUSPRIG/2001/Rajalingham_2000.htm, 3.39 pm 5.2.02]

Ross, J. and Ruhleder, K. (1993), Preparing IS Professionals for a Rapidly Changing World: The Challenge for IS Educators, Proceedings of the 1993 Conference on Computer Personnel Research, 379-384

Rushby, N. (1980), 'Debugging' in Meek B and Heath P (eds), Guide to Good Programming Practice', Ellis Horwood, UK

Sargent, P. (1990), "Give us the tools and we'll give you doorknobs", Times Higher Education Supplement, 30th March

Teo, T.S.H. and Tan, M. (1999), Spreadsheet development and 'what-if' analysis: quantitative versus qualitative errors, Accounting, Management and Information Technology, 9, 141-160

Spreadsheet Engineering: A Research Framework

From the Proceedings of the European Spreadsheet Risks Interest Group Symposium,
Cardiff, Wales, July 2002

Thomas A. Grossman
School of Business and Management, University of San Francisco,
San Francisco, California, USA 94117-1045
tagrossman@usfca.edu

ABSTRACT

Spreadsheet engineering adapts the lessons of software engineering to spreadsheets, providing eight principles as a framework for organizing spreadsheet programming recommendations. Spreadsheets raise issues inadequately addressed by software engineering. Spreadsheets are a powerful modeling language, allowing strategic rapid model change, and enabling exploratory modeling. Spreadsheets users learn slowly with experience because they focus on the problem domain not programming. The heterogeneity of spreadsheet users requires a taxonomy to guide recommendations. Deployment of best practices is difficult and merits research.

1. INTRODUCTION

People have programmed computers for at least five decades. Over this time there has emerged a field called "software engineering" that considers the myriad approaches people take—and should take—when they write computer programs. This knowledge includes journal publications describing theoretical research, laboratory experiments, field observations, and recommended practices, as well as industry wisdom codified in books and computer magazines.

Since a spreadsheet is nothing more than a computer programming tool, one hopes that some of the accumulated knowledge of software engineering is relevant to spreadsheets, and [Panko 2000a] recommends we start to adapt traditional programming techniques to spreadsheets. [Rajalingham et al 2000] take a step in this direction with design recommendations and a formal hierarchical tree technique. However, we are unaware of any systematic consideration of how software engineering principles could apply to spreadsheets.

The application of software engineering principles to spreadsheets—call this "*spreadsheet engineering*"—has the potential to increase the productivity of spreadsheet programmers, decrease the frequency and severity of spreadsheet errors, enhance spreadsheet maintainability over time, and actually be implemented by spreadsheet users.

The contributions of this paper are to present the principles of spreadsheet engineering; show how spreadsheets raise engineering questions not addressed by existing software engineering results (spreadsheets as a modeling language); use software engineering ideas to explain a vexing spreadsheet research result (spreadsheet programmer experience does not correlate with accuracy); propose that researchers require an improved taxonomy of spreadsheet users; and explain why the deployment of research results is itself a research topic.

2. SOFTWARE ENGINEERING

Software engineering is a large field with decades of history, countless journal articles, numerous textbooks and professional books, and a strong presence in university teaching (sixteen semester-length software engineering courses are listed in the University of Calgary 2001-2 Calendar). There is no satisfactory definition of software engineering—in fact, [Pfleeger 2001] doesn't even try to define it. [Institute for Information Technology 2000] provides several definitions, including this gem: "software engineering is what software engineers do". For our purposes, software engineering is concerned with all aspects of software creation.

Software engineering encompasses a broad range of issues. A partial list includes planning, design, requirements specification (figuring out what the software should do), prototyping, architecture, coding, error prevention, testing and debugging (identifying, finding and fixing errors), psychology of programming, organization and management of programmers, estimating time and cost, risk management, reliability, and lifecycle management.

Research results and extensive industry experience (see for example [McConnell 1993, 1996], [Pfleeger 2001]) show conclusively that good software engineering practices increase productivity, decrease the time needed to create software, reduce the number of errors, enhance the ability to maintain and update software over time, and can be widely deployed and used given skilled managers. A central lesson is that programmers must be cognizant of the *process* by which they create software. Superficially sensible but naïve approaches have been comprehensively shown to be ineffective and costly. Good software doesn't happen spontaneously: software must be engineered.

3. FROM SOFTWARE ENGINEERING TO SPREADSHEET ENGINEERING

Parallel to our definition of software engineering, we provide a working I provide definition of spreadsheet engineering: “Spreadsheet engineering is concerned with all aspects of creating spreadsheets.”

Decades of software engineering research and application has led to some important principles. These principles are independent of programming language, operating system, and computer hardware. Although they have not been verified on spreadsheets, we are optimistic these principles are valid for spreadsheet programs.

3.1. Eight Principles of Spreadsheet Engineering

Principle 1: Best practices can have large impact

Best practices can result in dramatically better results than naïve practices. McConnell [1996, p. 12] cites numerous studies showing productivity variations of “at least 10 to 1” for individual programmers with comparable levels of experience, and variations in the performance of entire teams “on the order of 3, 4, or 5 to 1”.

Principle 2: Lifecycle planning is important

There is extensive research that shows planning a software project leads to dramatic improvements in cost, time, and accuracy. However, some managers are unsympathetic to programmers who spend time on planning, and some programmers don’t plan because they can’t resist the urge to begin coding as soon as possible [McConnell 1996 p. 23].

Whether or not the programmer realizes it, every piece of software goes through a lifecycle, starting with the conception of the software and ending the last time the software is used. A key tool is a **lifecycle model** that “establishes the order in which a project specifies, prototypes, designs, implements, reviews, tests, and performs it other activities. It establishes the criteria that you use to determine whether to proceed from one task to the next” [McConnell 1996].

The lifecycle model chosen for a project defines the master plan for the project. The lifecycle model used has significant influence over the project’s success. Failure to choose a lifecycle model is itself a choice, often leading to *de facto* use of the undesirable “code-and-fix” model with unfortunate and predictable consequences.

Principle 3: A priori requirements specification is beneficial

It is easiest to build software when you know exactly what you want. However, specifying requirements in advance is difficult, in part because the act of programming teaches people about the problem they are trying to solve. Best practices depend on the degree to which the software requirements can be specified in advance, and make corresponding provision for learning and change during the programming process. A mismatch between the stability of the requirements and flexibility of the lifecycle model can lead to serious difficulties.

Principle 4: Predicting future use is important

Best practices vary with the predicted future use of the software. Software intended for one-time use by a domain expert should be constructed differently than software intended for repeat use by

multiple users with heterogeneous domain knowledge. Inaccurate prediction of future use can cause increased frustration, expense, and risk.

Principle 5: Design matters

Designing software before coding it improves cost, time and accuracy. Effective design requires application of a set of design principles. Some design principles apply to all programming languages, such as modularity and information hiding, and visual layout showing the logical structure of the program [McConnell 1993, 1996]. Some are specific to a language, such as use of formatting and color in a spreadsheet. Design decisions depend on knowledge of what the software will do, and how it will be used.

Principle 6: Best practices are situation-dependent

There are no one-size-fits-all best practices. There may be more than one "smart" approach to a particular challenge. No one method is best for all situations. Any proposed best practice should clearly state the situation(s) to which it applies.

Principle 7: Programming is a social, not an individual activity

There is a tendency to focus on the activities of individual programmers. However, programming is a social activity, and consideration must be given to the social unit [Weinberg 1998]. Standards and practices are transmitted and supported by social interactions. In addition, there is strong evidence for the benefits of programmers working together to create code and to read, review or inspect code [McConnell 1993, 1996, Panko 1999].

Principle 8: Deployment of best practices is difficult and consumes resources

Adoption of proven software engineering practices varies widely within and across firms [see Maguire 1994, McConnell 1999]. A key managerial and leadership challenge is getting programmers and their supervisors to adopt best practices. Many best practices require an upfront investment of time and resources that yields net savings in the future, necessitating a disciplined approach and careful management. Motivation, teamwork, selection and training are important considerations. The social environment is an important factor.

3.2. Spreadsheet Engineering Research Issues

Each of the principles in the previous section raises issues for spreadsheets that merit additional research, not least on whether these principles are indeed valid for spreadsheets. It seems desirable to interpret existing spreadsheet recommendations in light of these principles.

Theoretical and practical research on spreadsheet lifecycle models has high potential to generate useful insight and provide a structure to compare existing spreadsheet programming recommendations.

Spreadsheets raise certain issues that are inadequately addressed by software engineering. We discuss these issues and the spreadsheet engineering research implications in sections 4-7.

4. SPREADSHEETS AS A POWERFUL MODELING LANGUAGE

Spreadsheets are a uniquely powerful business modeling language that enable modeling that would be cumbersome or impossible in other languages. In particular, spreadsheets allow rapid model changes with strategic impact, and are a uniquely effective vehicle for exploratory modeling. These issues are uncommon in software engineering.

4.1. Spreadsheets for Rapid Model Changes with Strategic Impact

Spreadsheets allow rapid, near instantaneous modifications to existing models. In contrast to other computer languages, it is possible to make changes to the computer code (not changes in inputs to an application, but changes to the application itself) in real time or near real time. The speed advantage for custom analyses is so large that strategic advantage can be obtained.

[Schrage 2000] provides several examples. He describes negotiations between the government and investors to sell the assets of distressed savings and loans. The government undersecretary of the treasury—himself a former top investment banker—concedes that the government was "outspreadsheeted" by the investors who "would often bring their personal computers to the negotiations…and they could model practically every disputed issue instantly."

Schrage describes an investment banking team that developed dozens of spreadsheet models overnight in response to changing acquisition offers from multiple potential buyers. "Instant analyses turned into prototypes for instant counteroffers."

These examples of rapid model change fundamentally alter the ability of firms to compete: they are of strategic importance. It is difficult to imagine any other computer language allowing such a capability. More subtly, because end-users were able to do the modeling and programming, the delays and miscommunication attendant to non-end-user programming were avoided. The spreadsheet in the hands of an end-user programmer can function as a strategic weapon.

Implications for Spreadsheet Engineering Research

This class of programming—high stakes, high speed coding with strategic implications—appears to be absent from the software engineering literature. There are many interesting issues regarding this sort of programming, particularly how to design a spreadsheet to ensure flexibility and reduce the likelihood of errors.

4.2. Spreadsheets for Exploratory Modeling

Spreadsheet users sometimes start programming with only a vague idea of what they are doing. Software engineering wisdom suggests this is poor *programming* practice. However, this can be excellent *business* practice, because exploratory modeling teaches users about their business.

When performing exploratory modeling in a spreadsheet, the spreadsheet serves as a modeling tool to structure, explore, and understand a problem; it becomes a means for expressing one's ideas. Somewhere in the process of creating the spreadsheet, users realize something they didn't know before, recognize an insight that previously eluded them, articulate key issues to colleagues, reconceptualize their problem or even figure out how to solve it. Sometimes exploratory modeling helps them define the requirements of a computer program they then construct.

Field research [Sonntag and Grossman 1999], anecdotal evidence, and my observation of business students working on case assignments in spreadsheets indicate that exploratory modeling is a powerful and important use of spreadsheets. [Schrage 2000] provides a broad discussion of the business benefits of modeling and prototyping. [Powell 1995] provides guidelines for effective exploratory modeling.

Ethnographic research supports the value of exploratory spreadsheet modeling. [Nardi 1993, p. 83] indicates that spreadsheet users "discovered needs as they went along, and the interaction with the spreadsheet package directly supported this vital process." When discussing problem-solving, she argues that users need a means to figure out what they want, and therefore "the time-consuming struggle of iteratively and incrementally developing programs is a necessary and irreplaceable component of any end user programming activity".

Exploratory modeling is consistent with the management science literature on modeling and problem-solving, which emphasizes the importance of identifying the right problem [Evans 1991 is one example] and extracting insight from models [Geoffrion 1976]. Furthermore, [Ackoff 1981] distinguishes between an expedient "resolving", an optimal "solving", and a process-change "dissolving" of a problem. Exploratory modeling—in Ackoff's words "formulating the mess"—is central to the highly desirable third approach.

Exploratory modeling is in marked contrast to the traditional view of programming. Usually, "programming" suggests the use of a computer in a purposeful manner to accomplish a *specified task*, sometimes embodied in a requirements specification. In contrast, exploratory modeling is intended to *identify the task* that is to be accomplished. The power of exploratory modeling leads to a startling conclusion: There are times when it is actually desirable for an end-user to start programming with only a vague idea of what they are doing!

Be Careful with Artifacts

When someone performs exploratory modeling in a spreadsheet, it looks like they are programming in a spreadsheet. They are of course in some sense programming a spreadsheet, but they are not purposefully constructing a useful computer program. They are engaging in the intellectually demanding task of modeling, with the spreadsheet serving as a vehicle for expression.

During the modeling process, exploratory modelers learn much and benefit greatly. When they are done with their exploratory modeling, they find themselves in possession of an *artifact*: a spreadsheet. This spreadsheet artifact is the residue of their inchoate modeling process. This spreadsheet artifact is intimately connected to the powerful learning the user acquired during its creation. It contains many lessons for the circumspect.

Unfortunately, this spreadsheet artifact strongly resembles a purposeful computer program. It is tempting—sorely tempting—to use this artifact as the basis for further programming. However, this artifact is not suitable for programming, because it was not designed for that purpose. If it is misused in this fashion many difficulties are certain to arise.

Although it can be desirable for a user to start exploratory modeling in a spreadsheet with only a vague idea of what they are doing, it is risky to use the resulting spreadsheet as the foundation for a purposeful program. We hypothesize that the root cause of some end-user spreadsheet problems is misuse of the artifact of exploratory spreadsheet modeling, by making it the basis for a program intended to accomplish a specified task.

Implications for Spreadsheet Engineering Research

Exploratory modeling in spreadsheets is important and desirable. Spreadsheet engineering practices need to accommodate it, not fight it. Nardi [1993 p. 83] recommends that "rather than attempting to provide tools that avoid the need for incrementally working problems, it will be more fruitful to develop tools that make the struggle as short, attractive and productive as possible." In particular, it may be desirable to employ a distinct spreadsheet design step after exploratory modeling but before further programming.

Some spreadsheet programming projects start as modeling projects and evolve into a mixture of purposeful programming and continued modeling. These projects will experience substantial changes during development, and spreadsheet engineering practices must accommodate these changes during development. In these situations, flexible lifecycle models have advantages over less flexible models such as the traditional sequential "waterfall model" of software development.

5. THE PROBLEM OF EXPERIENCE

There is an old saying that "experience is the best teacher". Common sense suggests that in most human endeavors, experience leads to higher productivity and quality, and spreadsheets ought to be no different. Alas, this is not so.

There is experimental evidence that spreadsheet users do not seem to become more effective with experience. We propose a theoretical explanation that may aid in improving spreadsheet practice.

5.1. A Troubling Result: Experience Does Not Matter

Spreadsheet research has generated a troubling result: increased experience using spreadsheets does *not* correlate with increased quality. [Panko 2000b, 200c] summarizes three experimental studies on the role of experience on developing and auditing spreadsheets. They show no correlation, or no significant correlation between experience and quality. In a study of debugging models with seeded errors, "Expertise increased speed but did not reduce errors. Experts caught 57% of the errors while novices caught 55%." In a study of two programming tasks, "no difference in error rates among groups…experience made little difference." In a study of a programming task, experienced subjects showed moderate improvement over inexperienced subjects, but the results were not statistically significant.

5.2. Substantiation from Ethnographic Research

The experiments are substantiated by ethnographic research by Nardi [1993 p. 45] who indicates spreadsheet end-user programmers do learn slowly.

> "Spreadsheets allow users to perform useful work with a small investment of time and then to go on to more advanced levels of understanding as they are ready. In our research, we found that users often *add new programming concepts to their repertoire very slowly*, all the while being very productive spreadsheet users" (italics added).

5.3. Amateur and Professional Programmers

This strange phenomenon is consistent with spreadsheet users being "amateur" programmers. In 1971, Gerald Weinberg wrote a classic book called "The Psychology of Computer Programming". He makes an important distinction between amateur and professional

programmers [Weinberg 1998, p. 125]. He first describes how amateur programmers react to a programming challenge:

> "The amateur, being committed to the results of the particular program for his own purposes, is looking for a way to get the job done. If he runs into difficulty, all he wants is to surmount it—the manner of doing so is of little consequence."

Continuing, Weinberg describes how a professional has a different reaction:

> "Not so, however, for the professional. He may well be aware of numerous ways of circumnavigating the problem at hand…But his work does not stop there; it begins there. It begins because he must *understand* why he did not understand, in order that he may prepare himself for the programs he may someday write which will require that understanding."

Weinberg is suggesting that the task of increasing one's domain knowledge is a separate activity from increasing one's programming knowledge. The act of writing software to solve a problem is a guarantee of learning neither about the problem, nor about writing software; any learning is dependent on where the user focuses their attention.

The Key Distinction Between Amateur and Professional

Weinberg continues to define the key distinction between amateur and professional programmers:

> "The amateur, then, is learning about his *problem*, and any learning about programming he does may be a nice frill or may be a nasty impediment to him.
>
> "The professional, conversely, is learning about his *profession*—programming—and the problem being programmed is only one incidental step in his process of development."

These are significant differences in approach to programming. Weinberg's theory indicates a tradeoff between enhancing domain knowledge and enhancing programming knowledge.

5.4. The Troubling Result Explained

The experimental studies and ethnographic observation are consistent with the theoretical prediction that spreadsheet end-users, who by definition are interested in solving a problem, are unlikely to acquire professional spreadsheet programming skills with experience.

This seems to explain the troubling experimental results: experienced spreadsheet users are but amateur spreadsheet programmers. Although their experience presumably taught them much about their problems, it taught them little about spreadsheet programming.

Research Implications

Therefore, researchers and managers need to be cautious about interpreting the claims of self-described experienced spreadsheet programmers. We must distinguish between experience working in the spreadsheet environment, and experience studying spreadsheet programming. An instrument that measured spreadsheet programming ability would be a valuable contribution.

6. HETEROGENEITY OF SPREADSHEET PROGRAMMERS

Software engineering methodologies are valuable, and should be valuable when applied to spreadsheets. However, these methodologies are intended for a homogeneous group of professional computer programmers. Spreadsheet users are a heterogeneous group, and the techniques suited for one set of users will likely prove unsuitable for another set of users. To make useful recommendations we must classify spreadsheet users.

6.1. Existing Classifications of Spreadsheet Programmers

We are aware of two dichotomous classifications of spreadsheet users: end-user or non-end-user; and Weinberg's amateur or professional. (There are undoubtedly others, but they have not come to my attention.)

The end-user/non-end-user definition is essentially the programmer's *intention*; whether the program is written for the programmer or for someone else.

This definition is problematic because an individual is an end-user when writing a spreadsheet for his own use but is a non-end-user when writing for someone else's use. There is anecdotal evidence that end-user programmers regularly write spreadsheets for others in their workgroup. Worse, spreadsheets created for personal use are regularly transferred to other users.

The amateur/professional definition is concerned with the programmer's *attitude*: whether the programmer is focused on learning about the domain or learning about programming. Amateurs learn about programming slowly. Presumably, they are likely to make more mistakes and be less productive than professionals. The distinction between amateur and professional is not time spent programming, but time spent learning to be a better programmer.

The amateur/professional dichotomy is really a distribution. In between the extremes of the amateur who views programming as an impediment, and the professional who carefully studies every programming lesson and bug, lies a broad range of skills and interests.

6.2. Problems with Existing Classifications

There seems to be an inadequate vocabulary to describe important attributes of spreadsheet users. We observe a tendency to over-generalize, and to offer prescriptions that are appropriate only for a poorly-defined subset of users. This suggests that we require a richer language to classify people who use spreadsheets: we need a taxonomy of spreadsheet users.

Further evidence for the limitations of our current classification is a tendency to equate end-users and amateurs. This may have its roots in history. Before the personal computing revolution, corporate software was written solely by computer professionals because only they could access the mainframe computer. With the advent of PC's, anyone could program a computer. Computer professionals chose to use the faster, networked, more powerful mainframes while the end-users—amateurs all—were still denied access to the mainframes and used PC's. Thus, there was a time when virtually all end-users were amateurs who used PC's.

That time is past. Although today many end-users are indeed amateur programmers, there are many end-users who are skilled programmers. Some end-user programmers would undoubtedly qualify as professionals under Weinberg's definition. Some spreadsheet programmers in management consulting firms create spreadsheets for their own use, and then transfer them to

clients. They know many programming techniques for their narrow range of models, but it is unlikely their techniques would be as effective applied to different classes of models.

6.3. Towards a Taxonomy of Spreadsheet Programmers

We need a taxonomy of spreadsheet programmers that captures a rich set of user attributes. A useful taxonomy would help us understand the range of spreadsheet programming practice, and enable us to develop, test, and deploy different prescriptions for different members of the taxonomy. It would allow us to specify the situations where existing spreadsheet recommendations should be used, and avoided.

Such a taxonomy would likely capture the differences in programmer *activities* and *training*, not merely their *intentions* and *attitudes*. It might consider the user's awareness of and ability to apply principles 2 through 5 in section 3.1. Whatever taxonomy is developed, it needs to be firmly embedded in field research that examines what spreadsheet users actually do.

7. DEPLOYMENT OF SPREADSHEET ENGINEERING PRACTICES

Software engineering recommendations are based on an unstated assumption: the user desires to become a better programmer. Although the software engineering literature recognizes a range of choices in a particular situation regarding which practices to use, how to deploy selected practices, the best socio-cultural environment, and actions that managers should take and eschew, the alternatives only make sense when users are conscious of the programming choices they make, and are committed to becoming more effective programmers. Software engineering practices devised for professional programmers may not be relevant to amateurs.

If spreadsheet engineering research really does have significant potential to improve the practice of spreadsheet programmers, then researchers have a duty to see it gets used. It is not sufficient merely to show in the laboratory or in a few partner organizations that improvement is possible. Improved practices need to be actively deployed to spreadsheet users.

7.1. Spreadsheet Engineering Deployment Research Issues

Successful deployment of a spreadsheet engineering methodology is likely to be a significant challenge. Research into how to deploy best practices is as important as research on the best practices themselves.

Social issues seem particularly important in end-user programming environments [Nardi and Miller 1990, Nardi 1993], and exploitation of existing social structures may be essential to deployment efforts.

Organizations have a *de facto* spreadsheet culture. We hypothesize that in many organizations the spreadsheet culture misperceives the difficulty of spreadsheet programming, devalues thoughtful planning and design practices, provides few incentives to learn about spreadsheet programming, and provides powerful disincentives to open discussion of important spreadsheet errors.

We need research on how end-user spreadsheet programmers can be induced to adopt better practices, particularly in decentralized organizations where the programmers are domain experts with strong personalities and political power. One approach could be to motivate users and decision makers to treat spreadsheet programming with the same level of attention and

professionalism they treat the problem domain: that is, to “professionalize” the practice of spreadsheet programming.

It may be desirable to integrate research on deployment with research on best practices by explicitly developing practices that are easy to deploy. For example, tools that we can slip into users’ current practice will be easier to deploy. It might be beneficial to devise “second best” practices that are likely to be voluntarily adopted.

Deployment of systems based on strict policies and enforcement of standards [such as COBIT, Butler 2000, 2001] are expensive and probably require strong centralized management. They have high potential for effective application where powerful incentives or even legal sanctions apply (there are taxonomic research questions here), but general adoption seems unlikely.

7.2. Spreadsheet Asset Management

Large non-spreadsheet computer programs are generally treated as valuable assets. Like any valuable asset, their purpose and use is carefully considered before purchase or construction. A group of professionals designs, builds and tests the asset. The asset is carefully deployed. People are assigned to manage the asset and maintain it over time. For spreadsheets, even sizable, valuable spreadsheets, sound asset management structures seem to be uncommon.

Spreadsheet programs can indeed be valuable assets. A spreadsheet created in 3-months full-time work by a highly-paid professional has a cost-accounting value in excess of $25,000. Anecdotal evidence suggests spreadsheet assets are under-valued and inadequately managed—just imagine treating a company car like many users treat spreadsheets of equivalent value.

The issues of *spreadsheet asset management* merit research attention. What is the value of spreadsheet assets, and how are they being managed? Are valuable assets being squandered? How can the benefits of spreadsheet engineering techniques be communicated to management and users in a way that encourages their adoption? These benefits could include more rapid acquisition of domain knowledge, higher productivity, or *measurable* benefits from reduced error rates. Return-on-investment estimates of time and money invested in spreadsheet engineering training could prove persuasive.

7.3. The Limitations of Error Reduction as a Motivator for Change

Reformation of spreadsheet programming practice is often justified on the basis of error reduction. This argument is problematic because of the acceptance and apparent widespread satisfaction with current practice. Although there is widespread ignorance of the extent of spreadsheet errors and the risk of naïve spreadsheet practices, all software has errors—including any alternative to spreadsheets. Tolerance of spreadsheet errors is not necessarily foolish or even irrational; it is a matter of degree and of perceived risk.

We need research into managerial perceptions toward spreadsheet risks and errors. We need research into alternative motivators for the adoption of spreadsheet engineering techniques.

8. CONCLUSIONS

Spreadsheet engineering is concerned with the creation of spreadsheets. Principles of software engineering provide a useful starting point for spreadsheet engineering. It seems desirable to embed existing research on spreadsheet design processes in a spreadsheet engineering framework.

This will highlight gaps in our knowledge, and identify opportunities to adapt existing software engineering knowledge to spreadsheets, reducing the time necessary to develop and deploy improved spreadsheet practices.

Spreadsheet programmers are heterogeneous; including amateurs, professionals, end-users, exploratory modelers, and rapid-change aficionados, who work in diverse cultural and managerial environments. A taxonomy of users and their attributes is necessary to develop and deploy best practices.

9. ACKNOWLEDGMENTS

Stephen G. Powell made valuable suggestions on a draft of this paper. Comments by three anonymous referees improved the paper and provided inspiration for future work. This paper was written while the author was at the University of Calgary and was partially supported by Canada's NSERC, grant OGP0172794. All errors are the sole responsibility of the author.

10. REFERENCES

Ackoff, R. (1981), "The Art and Science of Mess Management", Interfaces 11(1): 20-26.

Butler, R. (2000), "The Subversive Spreadsheet", Presentation at INFORMS Miami conference, downloadable as INFORMS_1.PDF at http://www.ucalgary.ca/mg/grossman/grossman_srig.html , accessed April 30, 2002.

Butler, R. (2001), "Applying the Cobit Control Framework to Spreadsheet Developments", European Spreadsheet Risks Interest Group Symposium Proceedings, Amsterdam, pages 7-13.

Evans, J. (1991), Creative Thinking in the Decision and Management Sciences, South-Western.

Geoffrion, A. (1976), "The Purpose of Mathematical Programming is Insight, Not Numbers", Interfaces 7(1): 81-92.

Institute for Information Technology (2000), "Software Engineering Defined!?!", http://seg.iit.nrc.ca/English/sedefn/SEdefn.html, accessed 29 April 2002.

MaGuire, S. (1994), Debugging the Development Process, Microsoft Press.

McConnell, S. (1993), Code Complete: A Practical Handbook of Software Construction, Microsoft Press.

McConnell, S. (1996), Rapid Development: Taming Wild Software Schedules, Microsoft Press.

McConnell, S. (1999), After the Goldrush: Towards a Profession of Software Engineering, Microsoft Press.

Nardi B. and J. Miller (1990), "An Ethnographic Study of Distributed Problem Solving in Spreadsheet Development", Computer Supported Cooperative Work Proceedings: 197-208. ISBN 0-89791-402-3.

Nardi, B. (1993), A Small Matter of Programming: Perspectives on End User Computing, 1993, MIT Press.

Panko, R. (1999), "Applying Code Inspection to Spreadsheet Testing", Journal of Management Information Systems, 16(2): 159-176.

Panko (2000a), "What We Know About Spreadsheet Errors", http://panko.cba.hawaii.edu/ssr/Mypapers/whatknow.htm, accessed April 30, 2002.

Panko (2000b), "Errors in Spreadsheet Auditing Experiments", http://panko.cba.hawaii.edu/ssr/auditexp.htm, accessed April 30, 2002.

Panko (2000c), “Errors During Spreadsheet Development Experiments”, http://panko.cba.hawaii.edu/ssr/devexpt.htm, accessed April 30, 2002.

Pfleeger, S. (2001), Software Engineering Theory and Practice, 2nd Edition, Prentice Hall.

Powell, S. (1995), "The Teacher's Forum: Six Key Modeling Heuristics," Interfaces, 25(4): 114-125.

Rajalingham, K., D. Chadwick, B. Knight, D. Edwards (2000), “Quality control in spreadsheets: a software engineering-based approach to spreadsheet development”, Proceedings of the 33rd Hawaii International Conference on System Sciences, Maui, Hawaii.

Schrage, M. (2000), Serious Play: How the World’s Best Companies Simulate to Innovate, Harvard Business School Press.

Sonntag, C. and T. Grossman (1999), “End user modeling improves R&D management at AgrEvo Canada”, Interfaces 29(5): 132-142.

Weinberg, G. (1998), The Psychology of Computer Programming, Silver Anniversary Edition, Dorset House.

A Spreadsheet Auditing Tool Evaluated in an Industrial Context

Markus Clermont, Christian Hanin, Roland Mittermeir
Universität Klagenfurt
Universitätsstraße 65- 67
A-9020 Klagenfurt
Austria
mark@isys.uni-klu.ac.at

ABSTRACT

Amongst the large number of write-and-throw-away-spreadsheets developed for one-time use there is a rather neglected proportion of spreadsheets that are huge, periodically used, and submitted to regular update-cycles like any conventionally evolving valuable legacy application software. However, due to the very nature of spreadsheets, their evolution is particularly tricky and therefore error-prone.
In our strive to develop tools and methodologies to improve spreadsheet quality, we analysed consolidation spreadsheets of an internationally operating company for the errors they contain. The paper presents the results of the field audit, involving 78 spreadsheets with 60,446 non-empty cells. As a by-product, the study performed was also to validate our analysis tools in an industrial context.
The evaluated auditing tool offers the auditor a new view on the formula structure of the spreadsheet by grouping similar formulas into equivalence classes. Our auditing approach defines three similarity criteria between formulae, namely copy, logical and structural equivalence. To improve the visualization of large spreadsheets, equivalences and data dependencies are displayed in separated windows that are interlinked with the spreadsheet. The auditing approach helps to find irregularities in the geometrical pattern of similar formulas.

1 INTRODUCTION

Spreadsheets are a main factor contributing to the success of the personal computers. Today, they might be considered to be the most successful end-user programming tool. Each year, millions of spreadsheets are developed. Lots of them are small and used for one-time calculations, but there is a substantial number of spreadsheets that are large and complex.

These are usually strategically important and contain both large and complex calculations. These sheets might also be quite long-lived. Hence, undergo similar evolutionary steps as conventional software. In [Tampoe, 1996], spreadsheets are presented as strategic management information systems. Thus, erroneous spreadsheets, notably those long-living ones will have severe consequences.

The strategic spreadsheets we analysed generally consist of two parts: The first part is very large, but relatively uniform. It serves to gather data and to perform some quite simple calculations. This part can be spread out on very large areas of a sheet. It needs not to be contiguous, but it tends to be so. In the sheets we analysed, up to 20 columns and more than 200 rows are common in this part. The second part is much smaller, but contains more complex calculations. Examples are calculation of enterprise-specific financial ratios, time-series analysis or the generation of check-sums. While the first part confronts the auditor with a complexity ‘of size’, the complexity of the second part is due to a limited number of complex calculations.

Of course, one must not over-generalize from the sample of 78 sheets we analysed over a period of three months. But it seems fair to assume that any developer of a sheet that is repeatedly used strives to for an arrangement that is somehow related to the semantics of the sheet. Normally, this arrangement follows a well-understood business pattern. With large sheets, such business logic leads

to arrangements where data-entry cells, cells immediately dependent on these data entries used for preparatory operations, and cells performing the final modelling or analysis are allotted to distinct, well identifiable locations (to avoid confusion we avoid the term “area” at this moment) or laid out in a regular pattern.

Moreover, the sheets we analysed seem to be typical for sheets involved in financial or commercial applications. In [Filby, 1993] numerous applications of spreadsheets in science and engineering are presented. These spreadsheets are used in physics, chemistry and other sciences, because they are a more usable alternative to FORTRAN-programs and because they incorporate already the (graphical) representation of their result. As these spreadsheets specialize on complex calculations, we do only find the second part mentioned above. The data-entry portion is comparatively simple in these cases.

Our auditing methodology reduces the complexity of size by banking on regularities in the cell content. Similar cells are grouped into so-called logical equivalence classes. Cells that are in the same logical equivalence class are presented to the user by a single abstract unit, the logical area. When the logical areas are highlighted on the spreadsheet, the user can easily spot inconsistencies between the geometrical pattern of formula usage and the conceptual model they had in mind. Complex calculations that occur only in a few cells of the spreadsheet still have to be examined on a cell-by-cell level (c.f. [Panko, 1997]).

The rest of the paper is organized as follows: Section 2 points out the main sources of errors discovered in our field audit. In section 3 we briefly explain our auditing technique and present the toolkit used. Additionally we describe the reviewed spreadsheets and the context of their use. In section 4 the results of the field audit are presented and we try to categorize the revealed errors. Section 5 addresses the methodological issues involved with the experiment.

2 ERROR SOURCES

Indirectly, the ease of creating spreadsheet programs is the most important source of errors: Spreadsheet programs can be created without a great deal of IT-training and even complex models can be implemented by rather simple means.

The low level of the spreadsheet users IT-training will make them neglect important tasks like analysis, documentation and in-depth testing, as it seems that there is no direct relation between these tasks and the success of a spreadsheet program. [Nardi, 1990] states, that the spreadsheet is also an important modelling tool for the users. Thus, the spreadsheet program is quite often all in one: the modelling tool, the design and the implementation of an information system.

This procedure is in sharp contrast to the importance of spreadsheets for organizations. [Gable, 1991] analysed the importance of 400 spreadsheets for their organizations, and came to the conclusion, that more than 50% of them were considered to be very important. [Chan, 1996] interviewed more than 200 spreadsheet users on their estimation of the cost of an error in their spreadsheet. 4.6% estimated the potential damage is more than 1,000,000 USD. In [Panko, 2002] some drastic examples for spreadsheet errors that economically damaged the affected organization, are reported.

2.1 Complexity

Although spreadsheets are not very complex to create, the mechanism of absolute and relative cell references will rapidly lead to a high degree of complexity within them. Spreadsheet users are generally not aware of that fact. Thus, mistakes, that have been made anywhere in the underlying model, will be propagated.

The principle of locality, an important concept for reducing the complexity of software, is not part of the spreadsheet model, i.e. any other cell anywhere on the spreadsheet can freely access the result value of a certain cell. Hence, the effects of an error in an arbitrary cell will potentially influence one

or more results of the spreadsheet irrespective of their “distance” to the erroneous cell. Worse, the effect of an error might show at a different place than the error itself, thus further increasing the complexity of identifying faults.

There are techniques to reduce the complexity of spreadsheet programs, by forcing the spreadsheet user to build modular spreadsheets (see e.g. [Knight, 2000], [Janvrin, 2000], [Stadelmann, 1993], [Wilde, 1993]). However, these techniques are not widely used yet. In contrast to these techniques, we do not aim to change spreadsheet users. We suggest taking the sheets they developed on an as-is basis. We do assume, however, that even computing-laypersons do not spread out their calculations on the sheet in a random order. In contrast, we assume that they use the (two) dimensions of the sheet in an intelligent manner to floor plan the layout of their calculations.

2.2 Copy and Paste

Usually spreadsheets are created by defining a formula and then copying this formula into the cells were the same or a similar functionality is expected. The same formula tends to occur very often, but the geometric distances between these occurrences can be quite large.

Thus, the copy/paste mechanism is somehow similar to the use of subroutines (rather macros) in conventional software. However, there are some important differences that entail dangerous side effects:

- If the copied formula is erroneous, the error is replicated, too.
- Past the copy operation, the duplicated cells forget from where they originated.
- If an error is detected and corrected only at one place, all the other copies of this formula remain still erroneous.
- Error corrections might be done on the value level only, thus leading to incorrect sheets in future instantiations.

2.3 Error Correction

As in conventional software, we identified error correction as an important source of future errors in spreadsheets. Spreadsheet users tend to check their spreadsheets on the numerical level. When mismatches between their expectations and the shown result occur, they often fail to debug the formula. This might be considered to be too time consuming, because the real cause of the wrong value shown at the given cell is not obvious. Therefore, they just overwrite a formula with a constant value. As a consequence, the error is currently corrected and the current sheet shows correct computations. However, further changes to the spreadsheet will not be reflected in this cell. Thus, a new, latent error is introduced.

Again, we do not want to over-generalise. However, considering the training of the clerks working with these sheets, it is no wonder that they focus on the value domain of their sheets. Considering the value domain, we have to credit them with respect for highest diligence and care. Being no programmers though, they did not see that below this value domain there is a model domain (or “program domain”) expressed by the network of formulas tightly interwoven by linkages of references and data-flow. Therefore, the problem that their models are correct only, if these models are correct on the model (or program-) domain first, was something they have only gradually accepted during the time they worked with us.

2.4 Maintenance

A given long-living spreadsheet usually continues to evolve. As we already learned from conventional software [Parnas, 1994], software ages with maintenance. In order to keep up with evolving requirements, ongoing adjustments must take place. Changes in the environment of spreadsheet

programs, like new tax-rates or new organizational structures, will force the spreadsheet users to maintain the spreadsheet.

However, the lack of documentation makes it hard for spreadsheet authors to understand the effect of changing a single cell has on the rest of the spreadsheet. If the maintainer is not the original author, these problems are further aggravated. Maintainers do not know about the authors' conceptional model of the spreadsheet. Thus, they have to perform maintenance based on their assumptions. It is obvious that this procedure will blur the initial spreadsheet model and makes it 'age', as it is stated by [Parnas, 1994] for conventional software, quite rapidly.

Another common maintenance operation is the intended change of the functionality of a certain spreadsheet program, in order to make it applicable for problems that are similar to the original problem. Therefore, only those parts of the spreadsheet are modified, where changes are obviously needed. Other parts are not modified, which can entail misunderstandings and errors in further maintenance cycles.

Obviously, the actual spreadsheet development process does not support the high importance of spreadsheet programs. A methodical approach, thorough testing and sufficient documentation, steps common for raising the quality of conventional software, are hardly ever used in spreadsheet development. The short maintenance cycles and the lack of modularisation also promote the introduction and propagation of errors.

3 ORGANIZATIONAL ENVIRONMENT OF THE FIELD AUDIT

This section will introduce the auditing technique, the organizational environment of the audit, and the characteristics of the audited spreadsheets.

3.1 Auditing Technique

As already mentioned in section 2.4, misunderstandings regarding the spreadsheet model will make spreadsheet maintenance error prone. Further, testing of spreadsheets is complicated, as the internal logic is not clear to the tester. We developed an auditing technique to reveal the spreadsheet model by showing the occurrences of similar formulas throughout the spreadsheet. Thus, regular patterns, or irregularities can be spotted at first sight.

Irregularities generally do not indicate an error, but they indicate a dangerous spot that has to be checked, whereas regular patterns are a hint for a direct manifestation of a conceptual model on the spreadsheet. As effective auditing of spreadsheets is stated to be an expensive and time consuming task [Panko, 1997], our auditing technique will reduce the number of cells to be examined by finding the potentially dangerous areas and focussing the auditors' attention on these areas. Further, we offer another view on the conceptual model. It shows the data-flow, i.e. the dependencies, between these regular areas.

By understanding the abstract representation our tool provides, the auditor can comprehend the architecture of the spreadsheet. Thus, error correction and maintenance are supported, as the maintainer is aware of regular patterns of formula-occurrences. This helps in comprehending sheets originally written by others.

Symptoms of errors are often erroneously corrected by overwriting the correct formula with a constant value or another formula. In these cases, the problem is only aggravated, because the formula just showing an incorrect value due to an error in another cell is destroyed by this pseudo-corrective act in the value domain. As auditing spreadsheets by finding irregularities is not based on symptoms, but on causes of errors, correction can be focussed and is thus easier to perform.

Our technique identifies regular structures in the spreadsheet. These regular structures, so called logical equivalence classes, are sets of similar cells. These similar cells do not have to be neighbours, but we noticed that on large sheets

- They are either neighbours on the layout, or
- They are distributed in a regular pattern, or
- Their occurrence is limited to a certain area of the spreadsheet

Of course, none of these points need to be the case. But for the majority of logical equivalence classes at least one of these properties applies.

Above, we defined the logical equivalence class to be a set of similar cells. The similarity is defined by comparing the formulas. We consider the following three kinds of equivalence classes:

1. **Copy-Equivalence** exists, if 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).
2. **Logical- Equivalence** exists, if the formulas differ only in constant values and absolute references
3. **Structural- Equivalence** exists, if the formulas consist of the same operators in the same order, but the operators may be applied to different arguments.

By comparing the partition of cells into logical equivalence classes with their geometric distribution on the spreadsheet, inconsistencies can be easily spotted. E.g. if a set of cells in a column is copy-equivalent, but there is one cell interspersed that contains a different formula or a constant, this indicates an inconsistency that has to be further investigated.

3.2 The Toolkit

In order to support the auditing process we developed a toolkit that automatically performs the partitioning into equivalence classes. The toolkit consists of three main parts: A structure browser (see Figure 1) to show the decomposition of the spreadsheet into equivalence classes, a dependency viewer that displays the data flow graph between these dependencies, and the spreadsheet itself giving feedback to the auditor by highlighting the cells that are in the equivalence class that is currently selected in the structure browser.

The structure browser uses the equivalence class hierarchy (see Figure 2) to give a hierarchic view. As the auditors are able to expand and collapse the nodes in the structure browser they can zoom into certain equivalence classes, whilst viewing the remaining nodes on a higher level of abstraction. Only those nodes that are visible in the structure browser are displayed in the dependency viewer.

As we used only an α-Version of our tool for the audit, the technical skills of the auditor were highly needed. The integration between the dependency viewer and the structure browser was rather rudimentary, by generating files in the structure browser and displaying them with the free graph layout software Dotty (see [Ganser, 1999]). In the subsequent versions of our auditing tool we aim for a tighter integration between dependency viewer, structure browser and spreadsheet.

3.3 Organizational Environment

Auditing was performed from April until August 2001 by a computer-science student in the sixth semester. The auditor was assigned to the accounting department of an international cooperation with headquarters in Vienna where he could work desk-to-desk with the spreadsheet producers. The contact with the tool developers was by e-mail and by regular visits. He examined three voluminous Excel-workbooks (see section 3.4) that are mainly used for consolidation. The three workbooks consisted of 78 worksheets, with 60,446 non-empty cells.

The identified errors were coarsely categorized by their immediate impact into qualitative and quantitative errors (see [Teo, 2000]), and by their origin into the following categories (see [Ayalew, 2000]):

- Constant instead of formula
- Constant instead of reference
- Reference to empty cell
- Formula copied too far
- Other

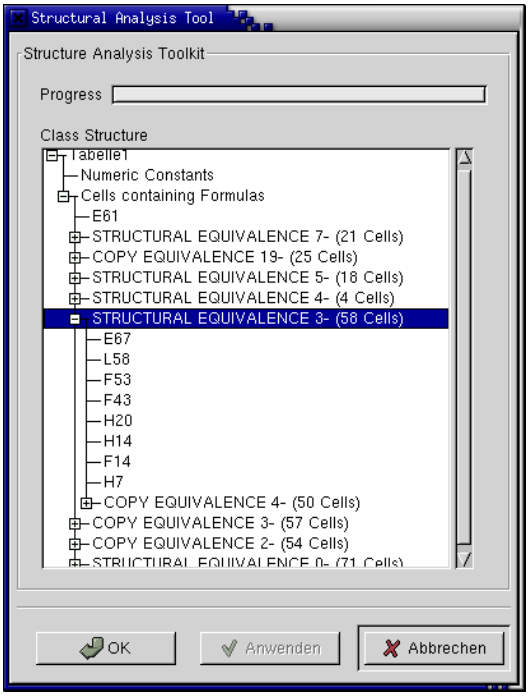


Figure 1: The Structure Browser with example-data

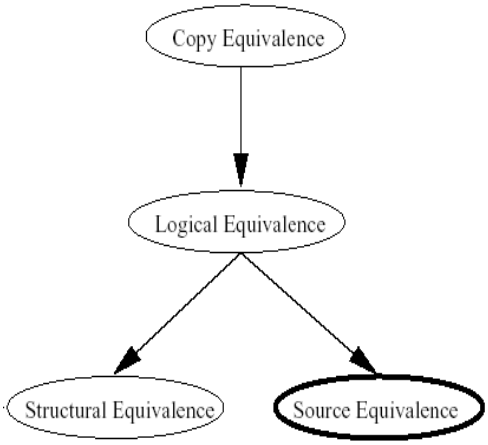


Figure 2: Relevant portion of the partial order between logical equivalence classes.

The consequence of a quantitative error is an erroneous result of a cell on the spreadsheet, i.e. a wrong result in the value domain. This does not necessarily relate to an error within the cell that contains the quantitatively erroneous formula. As already mentioned above, the error and the symptom of the error can turn up in different cells.

In contrast, qualitative errors will not immediately entail a wrong result in the value of any cell. However, they are (potential) errors in the model. When maintenance is performed, these qualitative errors usually turn into quantitative errors, i.e. somewhere on the spreadsheet a corrupted value will be displayed. An example for a common qualitative error is an erroneous expression in one branch of an if-statement in a certain cell. As long as the erroneous branch is not activated, there is no symptom of fault for this cell.

3.4 Auditing Process

Before the audit started, the auditor, who had only little bookkeeping experience, discussed the basic idea and functionality of each workbook with the respective author. Additionally, the author was interviewed about the lifespan of the workbook, the usual maintenance cycle and the number of users.

Then, for each spreadsheet in the workbook, the following characteristics were documented: Dimension, Number of occupied cells, Number of formulas, constants and literals. At first, the

correctness of the displayed values was checked. Special attention was put on wrong sums, wrong formatting and errors that were reported by Excel.
After these routine checks in the value domain, the toolkit described in section 3.2 was applied. The so discovered irregularities where then discussed with the spreadsheet authors, to find out, if the detected irregularities were deliberately introduced or whether they have to be corrected and counted in the error statistics.

Thus, the auditor had a lot of discussion with the domain specialists who created the spreadsheets. No error was documented that was not verified by the spreadsheet creator. The identified errors were collected in an error database. For each error we gathered information about the location, the kind of error, and its impact. Additionally, a short description was also stored.

As an error can be multiplied by copy and paste operations, we distinguish between errors and error classes. Copy-equivalent erroneous formulas are counted as one error-class. Hence, the error-class corresponds to the unique source of an error that can be copied into several cells. The term error is used to count each of the error-instances within the respective error class. Thus, each error represents an erroneous cell.

3.5 Examined Spreadsheets

The audit examined three large excel workbooks. Each of them was used to gather data from various departments of the company and to calculate different financial ratios at the corporate level. These financial ratios are an important base for strategic decisions. The workbooks analysed served the following purpose:

- **RAT-2001** calculates a financial statement. Data is aggregated from sub-sheets that correspond to the enterprise's organization. Hence, there are worksheets for different business-units (BU) and corporate sectors. These worksheets are aggregated to calculate the financial statement of each division. The spreadsheet has been in use for one year so far. There is extensive maintenance each month. The company's annual budget processed by these spreadsheets is about € 150,000,000.

- **TP-Report** was in use for three months when we examined it. The lifespan of the spreadsheet was considered to be unlimited. When audited, the author was the only user. But it was planned to delegate maintenance of particular worksheets to other employees. The sheet accumulates data from four other workbooks that are maintained by four different persons. During our study the workbook has been fundamentally changed, so we re-audited it. In the results we only mention the latest version audited.

- **AB-Market** performs material costs analysis. It is in use since 1999 and modified each year, before budgeting is done. A copy of the workbook is sent to each branch office where its input cells are filled in by at most three employees. The completed/updated workbooks are sent to the author again, who merges the copies into a single workbook. The data obtained by this procedure is used to analyse cost of raw material of the various factories. For the analysis, additional information, such as current and forecasted volume, costs, price per unit, and average prices are added to the workbook. This information is extracted from the companies SAP-based information system. The workbook calculates a budget target for each factory that can be compared to the planned budget. The calculated budgets' values are about €13,000,000 each.

4 RESULTS

Concerning error statistics, the results we obtained correspond to the findings of earlier studies and the reports of practitioners (see [Panko, 2002], [Butler, 2000]). The overall error rate was 3.03% of the non-empty cells. However, we did not find any tremendous erroneous result values that might have

had severe negative effects on the company. What we found though was a very high number of qualitative errors with the potential to become quantitative errors in the next (or future) maintenance cycle(s).

Thus, the numerical test that each workbook undergoes after each round of modifications becomes more difficult, and, as we argued above, the increasing number of "corrected" errors tends to introduce more qualitative errors in the model (see section 3.3). This vicious circle cannot be interrupted without corrections of the spreadsheet model.

4.1 Overview of Results

In 78 audited spreadsheets 109 error classes with 1832 occurrences were identified (see table 1). As the workbooks have usually consisted of similar spreadsheets, the occurrence of one error class is not limited to one spreadsheet. We identified several error classes that were copied into different spreadsheets of the same workbook.

The workbook TP-Report was still under construction when our study finished and so many of the identified problems were immediately corrected. This explains, so many error classes were detected in this workbook. The workbook AB-Market has been re-designed a short time before our audit took place. Hence, there was only a small amount of errors in the model.

The distribution of errors in the audited workbooks is given in absolute numbers in Table 1, whereas ´Table 2 gives the relative distribution with percent-values.

Workbook	**#Cells**	**#Occupied**	**#Formula**	**#Literals**	**#CE**	**#Error Classes**	**#Errors**
RAT-2001	56,485	19,444	12,382	7,062	814	21	257
TP-Report	69,835	23,502	16,873	6,629	950	83	1,561
AB-Market	66,385	17,500	7,174	10,326	95	5	14
Total	***192,705***	***60,446***	***36,429***	***24,017***	***1,859***	***109***	***1,832***

Table 1: Error Distribution, absolute

By classifying the errors and error classes into quantitative and qualitative errors, we obtained the distribution given in Table 3. The classification into the error-categories listed in section 3.3 is given in Table 4. The category *Others* consists of a wide diversity of error classes with patterns more or less unique for the individual instances.

Workbook	**#Cells**	**#Occ.**	**#Formula**	**#Literals**	**CE/Formula**	**#Error Classes**	**#Errors**
RAT-2001	56,485	34%	64%	36%	6.6%	21	1,3%
TP-Report	69,835	34%	72%	28%	5.6%	83	6,7%
AB-Market	66,385	26%	41%	59%	1.3%	5	0,08%
Total	***192,705***	***31.37%***	***60.27%***	***39.73%***	***5.1%***	***109***	***3.03%***

Table 2: Error Distribution, relative (#Errors is given relative to occupied cells)

Workbook	**Category**	**Error Classes**	**Errors**
RAT-2001	Qualitative	7	84
	Quantitative	14	183
TP-Report	Qualitative	73	1503
	Quantitative	10	58
AB-Market	Qualitatve	5	14
	Quantitative	0	0
Total	***Qualitative***	***85***	***1591***
	Quantitative	***24***	***241***

Table 3: Error classification into qualitative and quantitative errors

In order to check the effectiveness of the auditing technique, we calculated the Copy-Equivalence to Formula ratio, i.e. the average size of each copy equivalence class. In the average, each copy-equivalence class contains 5.1 formulas. Thus, only every fifth formula cell of the spreadsheet had to be checked in detail. Of course, this measure is blurred, as there are certain formulas, e.g. check-sums or other validation formulas, that occur only once, whilst others occur more than 20 times. For multiple occurrences of the same formula it had only to be checked, if they are used in the right place.
The frequency of occurrence of error classes relative to copy-equivalent classes is obviously correlated to the frequency of errors relative to formulas. This seems to support our assumption that errors are likely to be multiplied by copy & paste. However, as it is shown by Table 5, the workbook *AB-Market* does not follow this trend. We argue that this is because of the 'youth' of this workbook. The errors detected seem to be mainly in checksums and thus, not copied over many cells.

Error Category	**Error Classes**	**Errors**
Constant instead of formula	16	1222
Constant instead of reference	8	78
Reference to empty cell	8	78
Formula copied to far	24	215
Other	53	239

Table 4: Error distribution by error category

Workbook	**#Formula**	**#CE**	**#Error Classes**	**CE/Formula**	**Error Classes / CE**	**Errors / Formula**
RAT-2001	12382	811	21	6.6%	2,6%	2,07%
TP-Report	16873	950	83	5.6%	8,7%	9,25%
AB-Market	7174	95	5	1.3%	5,2%	0,19%
Total	***36429***	***1859***	***109***	***5.1%***	***5,9%***	***5,02%***

Table 5: Error Class Distribution, relative to copy-equivalence classes

5 TOOL ASSESSMENT

In spite of the analysis of the quality of strategic spreadsheets in use in our partner company, we were interested in evaluating the approach we developed for analysing spreadsheet quality. As spreadsheet users are application experts, we do not want to put too heavy a burden on them by requiring to switch from their "culture" as application experts to the "culture" of professional software developers. Nevertheless, they act as professional software developers when writing and maintaining long-living spreadsheets.

To assess our auditing technique's effectiveness, one has to recognise that there are two dimensions of freedom to be considered: The number of actual errors in the sheets available and the degree to which such errors are identified, and the effort needed to find those errors.

Obviously, testing and other conventional forms of software quality assurance can never demonstrate that the artefact analysed is faultless. Testing can only show that it finds faults. In our case, the auditor first analysed the sheets on the value dimension and found extremely few errors. This can be taken as indicator of the general high quality of the sheets. The ones he caught, though, can be taken as evidence for his careful checking and sufficiently mastering the application area. Looking on the model dimension, however, he found an overall error rate of 3,03 %. This not only meets our expectations, it is also consistent with results from other studies [Panko, 2000], [Panko, 1997b].

The second aspect is efficiency. The auditor who was no domain expert, stayed for 4 months at the company and actually spent 10 weeks on the audit. Hence, the examination of totally 60.446 cells was done in ten weeks by somebody who is not a domain expert. Of course, the errors identified were

discussed with the sheets' authors, and documentation work had to be done. This gives an average inspection rate of 1208 cells per day.

Compared to other approaches (see [Panko, 1997]) this is rather high. Hence, we claim that the approach is worthwhile to follow at least for those portions of sheets, where high regularity is to be assumed and that complexity of size is well addressed. The structural complexity, however, is still an issue warranting further investigations.

6 DISCUSSION

The main task of the audit was twofold. On the face value, our industry partner wanted to have the companies spreadsheet audited (To be honest: Before we started, they were convinced that we would not find anything!). We, on the other hand wanted to assess the feasibility and effectiveness of the approach to audit spreadsheets on the basis of visualization by logical equivalence classes.

Concerning the first aspect, we might say that the quality of the company's spreadsheet was surprisingly good at first sight. The audit did not reveal spectacular wrong results. This might be due to the fact, that the spreadsheets are properly tested. However, they test only in the value domain and the correction on the value level made the spreadsheet model inconsistent. This bears the danger of spectacular errors to come up in future evolution steps. However, the audit still discovered 241 quantitative errors in the spreadsheets.

The company's representatives were very concerned of the audit's result. They stated that better spreadsheet development practices are going to be introduced. The representatives were also interested in guidelines to decide, whether a specific application should be realized by a spreadsheet or by a database application. One of the suggested improvements was better documentation and the application of systematic testing and auditing approaches.

The efficiency and performance of testing can be increased by use of a standardized auditing or testing methodology, as described in [Rothermel, 2000] or in [Ayalew, 2002]. The efficiency can be further increased by model visualization (see [Mittermeir, 2002]).

Insufficient documentation turned out to be the main cause of errors. Thus, we are currently working on guidelines for the documentation of spreadsheets. The lack of understanding due to missing documentation can even make some spreadsheets useless, if the maintainer leaves the company. Better understanding can be gained either by decreasing the overall complexity of the spreadsheet with design restrictions (see [Knight, 2000], [Isakowitz, 1995], [Wilde, 1993]), by giving a more comprehensive description of the spreadsheet (see [Paine, 1997], [Stadelmann, 1993]) or by visualizing the logical structure (see [Sajaniemi, 2000], [Chan, 2000], [Mittermeir, 2002]).

7 FUTURE WORK

Currently we are improving our auditing tool by a seamless integration of the dependency viewer. We aim to place it into one of the next releases of the open-source spreadsheet system *Gnumeric*. Our plans to integrate the toolkit with *Excel* are currently stalled, as we do not have access to the excel-formula-parser, while comparing parse-trees is a main issue of our toolkit.

We aim to support the auditing of large spreadsheets by adding further abstraction mechanisms to our approach. Among other things, we suggest to find groups of similar cells with similar neighbours and group them into semantic classes. Again, these semantic classes can be used for spotting irregularities in the spreadsheet.

8 CONCLUSION

This paper presents an auditing toolkit for assessing the correctness of large spreadsheets. The tool helps to identify irregularities in the spatial distribution of similar formulas. An assessment in an industrial context proved to be quite encouraging. It helped to analyse 78 spreadsheets, amongst them 62% contained errors. The cell error rate was 3.03 %. For the auditing itself, 4 person-months have been spent.

It turned out that the toolkit is suitable for auditing spreadsheets with large uniform or regular blocks by reducing the complexity of size. The auditors attention is focused to those cells were the regularity of formula occurrences is interrupted.

The main error sources we identified were the lack of documentation, maintenance and error corrections that were not consistent with the spreadsheet's internal logic. Thus, further ways for supporting spreadsheet comprehension are called for.

9 REFERENCES

Yirsaw Ayalew (2001). *Spreadsheet Testing Using Interval Analysis*. PhD thesis, Universität Klagenfurt, Universitätsstrasse 65–67, A-9020 Klagenfurt, Austria.

Yirsaw Ayalew, Markus Clermont, and Roland Mittermeir (2000). *Detecting Errors in Spreadsheets*. In Spreadsheet Risks, Audit and Development Methods, (1):51-62

Ray Butler (2000). Is This Spreadsheet a Tax Evader ? *How H. M. Customs & Excise Test Spreadsheet Applications*. In Proceedings of the 33rd Hawaii International Conference on System Sciences - 2000, (33).

Hock Chuan Chan and Ying Chen (2000). *Visual Checking of Spreadsheets.* In Spreadsheet Risks, Audit and Development Methods, (1):75-85

Yolande E. Chan and Veda C. Storey (1996). *The Use of Spreadsheets in Organizations: Determinants and Consequences*. In Information & Management, 31:119-134.

Gordon Filby ed. (1998). *Spreadsheets in Science and Engineering*. Springer, Berlin, Heidelberg.

Emden R. Ganser and Stephen C. North (1999). *An Open Graph Visualization System and its Applications to Software Engineering.* Software Practice and Experience, 1-5 (1999)

Thomas Isakowitz, Shimon Shocken, and Henry C. Lucas (1995). *Toward a Logical/Physical Theory of Spreadsheet Modelling*. ACM Transactions on Information Systems, 13(1):1–37.

Diane Janvrin and Joline Morrison (2000). *Using a Structured Design Approach to Reduce Risks in End User Spreadsheet Development.* Information and Management, 37:1-12, 2000.

Brian Knight, David Chadwick, Kamalesen Rajalingham (2000). *A Structured Methodology for Spreadsheet-Modelling.* In Spreadsheet Risks, Audit and Development Methods, (1):51- 50.

Roland Mittermeir, Markus Clermont, and Yirsaw Ayalew (2002). User Centered Approaches for Improving Spreadsheet Quality. Technical Report TR-ISYS-MCA-1, Institut für Informatik-Systeme, Universität Klagenfurt.

Bonnie A. Nardi and James R. Miller (1990). *The Spreadsheet Interface: A Basis for End User Programming*. Technical Report HPL-90-08, HP Software Technology Laboratory, March 1990.

Jocelyn Paine. *MODEL MASTER: Making Spreadsheets Safe*. In Proceedings of CALECO97. CTI, 1997.

Raymond Panko (1997). *Applying Code Inspection to Spreadsheet Testing*. Working Paper, November 1997

Raymond Panko and Richard Halverson (1997b). *Are Two Heads Better than One? (At Reducing Errors in Spreadsheet Modelling)*. Office Systems Research Journal, 1997.

Raymond Panko (2002). *Spreadsheet -Research Homepage*. http://www.panko.com, visited 24.4.2002

David Lorge Parnas (1994). *Software Aging*. In Proceedings of the 16th international conference on Software Engineering, Volume 16, pages 279–287. IEEE.

Karen Rothermel, Curtis Cook, Margaret Burnett, Justin Schonfeld, T. Green, and Gregg Rothermel (2000). *WYSIWYT Testing in the Spreadsheet Paradigm: An Empirical Evaluation*. In ICSE 2000 Proceedings, pages 230–239.

Jorma Sajaniemi. *Modelling spreadsheet audit: A Rigorous Approach to Automatic Visualization* (2000). Journal of Visual Languages and Computing, 11(1):49–82.

Marc Stadelmann (1993). *A Spreadsheet Based on Constraints*. In Proceedings of the sixth annual ACM symposium on User Interface Software and Technology. pages 217- 224

Martin Tampoe and Bernard Taylor (1996). *Strategy Software: Exploring its Potential*. Long Range Planning, 29(2):239-245.

Thompson Teo and Margaret Tan (2000). *Spreadsheet Development and 'What-if' Analysis: Quantitative versus Qualitative Errors*. Accounting Management And Information Technologies, 9:141–160.

Nicolas P. Wilde, 1993. *A WYSIWYC (What You See Is What You Compute) Spreadsheet*. In Proceedings of the 1993 Symposium on Visual Languages, pages 72-76.

MANAGEMENT SUMMARIES

All papers in this section have been submitted by practitioners or professionals working in the fields of education, audit, IT, business, statistics or accountancy. Papers included are those deemed to provide a useful contribution to raising awareness of spreadsheet risks and improving practitioner understanding on how to deal with them.

EuSpRIG Talk: The Subversive Spreadsheet
Brian Knight and David Chadwick , University of Greenwich

Losing at Spreadsheet Roulette'
Ray Butler, HM Customs & Excise, UK '

A Typical Spreadsheet Audit Approach
Grenville Croll, formerly of Andersen's

Spreadsheet systems are the most widely used and the most popular end user systems. Hence, spreadsheets (we might refer to them as "spreadsheet programs") are an important basis for far reaching decisions in almost any field of a modern society. Studies on the quality of spreadsheets and spreadsheet based decisions show, however, that there is a substantial divergence between significance and care in this area.

Yirsaw Ayalew, Markus Clermont, Roland T. Mittermeir
Detecting Errors In Spreadsheets , EuSpRIG 2000

STOP THAT SUBVERSIVE SPREADSHEET !

David Chadwick
and
Grenville Croll

European Spreadsheet Risks Information Group

Todays Presentation

EuSpRIG
What Is The Problem?
Development Methods
Audit Products
Model Review

European Spreadsheet Risks Information Group

Founded March 1999
Ray Butler (HMCE) , David Chadwick (UoG), Patrick Cleary (UoW).

Symposia: July 2000(Greenwich), July 2001 (Amsterdam), July 2002 (Cardiff)
ISACA (2000/2001 sponsor), PWC (2001 sponsor)
BCS IRMA, KPMG, Anderson's
Universities of Hawaii, Singapore, Klagenfurt, Calgary, Amsterdam
Operis (OAK), Southern Cross Software (Spreadsheet Detective), HMCustoms (SPACE)

www.eusprig.org

WHAT IS THE PROBLEM?

"The use of spreadsheets in business is a little like Christmas for children. They are too excited to get on with the game to read or think about the 'rules' which are generally boring and not sexy"

David Finch Head of Internal Audit,Superdrug

WHAT IS THE PROBLEM?

'There is a huge body of evidence that errors in spreadsheet applications and models are alarmingly common (some authorities, with justification cite spreadsheets containing errors as the norm rather than the exception)' **Ray Butler HMCE**

'Endusers are putting their companies at risk by setting up spreadsheets without realising that this demands the discipline of traditional programming. Our findings are disturbing, but they are not really surprising, as 78% of models had no formal quality assurance to ensure they were built to specified requirements and were fit for purpose'
KPMG in Computer Weekly

'While it is true that databases and spreadsheets provide a safe processing environment ... it is still possible to make errors in logic. In fact, such errors maybe more difficult to detect than ... if a procedural programming language employed'
Computer Auditing' Chambers & Court

TEACHING ACCOUNTING
(A.Hawker U. of Birmingham)

Accuracy When Large Numbers of Significant Figures?
Declining balance function in Excel calculates and uses a factor which is accurate to three decimal places. So, if if an asset worth millions of pounds, the final few figures may be incorrect.

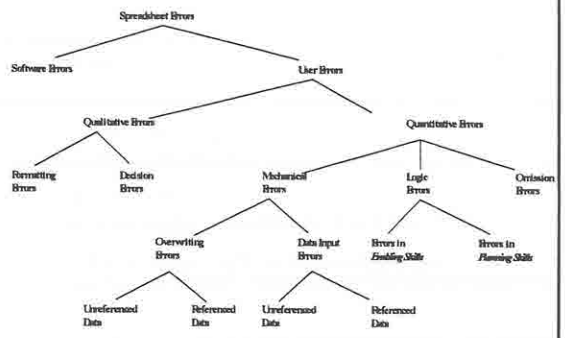
Depreciation
No problem if user converts results to round sums. Spreadsheets expected to deliver precision so anyone who adds up depreciation values over the life of an asset, and expects that the salvage value will always equate to the initial value minus the accumulated depreciation, could be in for a shock.

Misleading or ineffective "help".
Help screens advise that annual interest rates can be converted to monthly simply by dividing by 12. This advice ignores effective v. nominal interest rates. It accords with American practice, but causes problems if students replicate a loan repayment table issued by a UK bank. UK practice is to use compounded monthly interest). Divide annual rate by 12, and results will be too high.

Ray Panko: Field Audits of Real Spreadsheets

Study	Year	Spreadsheets	%w Errors
Davies & Ikin (1)	1987	19	21%
Butler (2)	1992	273	11%
Hicks	1995	1	100%
Coopers & Lybrand (3)	1997	23	91%
KPMG (4)	1997	22	91%
Lukasik	1998	2	100%
Butler (2)	2000	7	86%
Overall	NA	367	24%
			(5)
1997 and Later	NA	54	91%

NR = Not reported
(1) "Serious errors"
(2) Only reported errors large enough to demand additional tax payments
(3) Spreadsheets off by at least 5%
(4) "Major errors"
(5) Weighted average



Chadwick et al: Taxonomy of spreadsheet errors.

DEVELOPMENT

METHODS

"We nearly always find that the modellers have no formal training in good modelling techniques, and that their organisations do not even have the most rudimentary internal modelling standards"

Barry Pettifor,
Director of Spreadsheet Assurance Services
PricewaterhouseCoopers

Development Methods

Life-Cycle Stages
Visual Aids
Programming Languages
Model Reviews – see later

Univ. of Greenwich: R.A.D.A.R Spreadsheet Life-Cycle

Requirements	**What is the purpose of the spreadsheet. ?** **Which staff are to be consulted ?** **How long to do? What cost to do?**
Analysis	**Gather data : interviews, documents.** **Define Row/Column Titles,** **Define Modules: Data, Simple and Complex function** **Identify Title Links** **Create functions** **Create lookup data modules** **Identify Data Links between modules**
Design	**Create final layout** **Populate with data** **Test** **Create User-Guide**
Acceptance	**Train user with User-Guide** **Test with the user/client** **Handover to client.**
Review	**Let it operate for a while.** **Does it satisfy the client - if not then start again**

Inputs	New Quantity following income change	
	Old quantity before income change	
	New real incomes	
	Old real incomes	
Results	% change demand	=IF(ISERR(C1/C2-1),"",C1/C2-1)
	% change in real incomes	=IF(ISERR(C3/C4-1),"",C3/C4-1)
	income elasticity of demand	=IF(ISERR(C6/C7),"",C6/C7)
		=IF(C6="","",IF(C6>0,"normal","inferior"))

Attributes <

new_quantity, old_quantity, new_real_income, old_real_income, demand_change
real_income_change, income_elasticity, good_type

>Where

demand_change = new_quantity / old_quantity – 1
and real_income_change = new_real_income / old_real_income – 1
and income_elasticity = demand_change / real_income_change
and good_type =
if (income_elasticity > 0,
"normal",
"inferior""
)

Modelsmarter
Jocelyn Paine Univ. of Bristol

AUDIT

PRODUCTS

"The presence of a spreadsheet application in an accounting system can subvert all the controls in all other parts of that system".

Ray Butler , HM Customs & Excise

Audit Tools

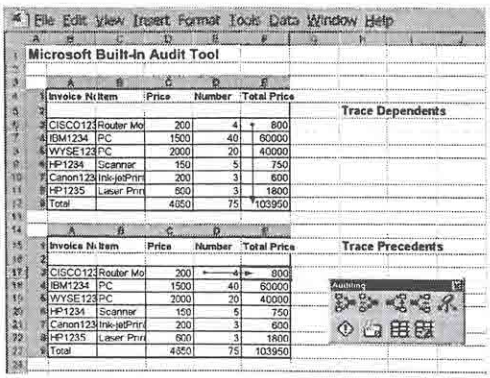
Microsoft Auditing Tool

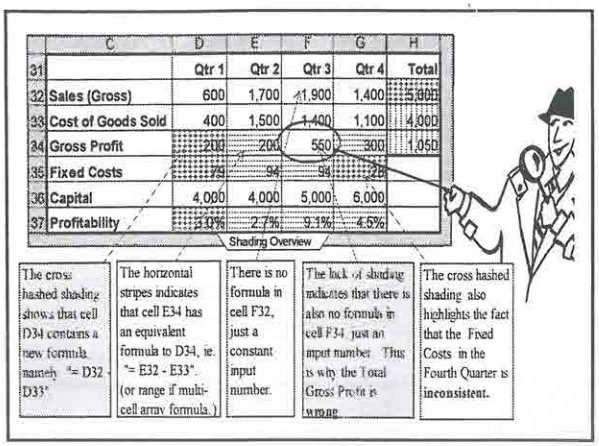
Spreadsheet Professional

Spreadsheet Detective

(OAK) Operis Analysis Kit

SPACE (Spreadsheet Audit for Customs & Excise)





Tools for Audit

SpACE Spreadsheet Auditing from Customs & Excise

- Customs & Excise's own product
- Risk assessment and testing methodology
- Software - add-in for Excel
- Training
- Tests *any* file Excel can open
- Works on copy file

Losing at Spreadsheet Roulette

Some Recent Cases

Ray Butler, CISA
Computer Audit Service,
H M Customs and Excise

HM Customs and Excise
Business Services and Taxes

To Remind Us Why We're Here....

- 3 examples of loss / damage
- Cause / impact / detection / prevention
- Some names changed to protect the guilty

The Examples

- Allied Irish Bank / Allfirst
- A School "Somewhere in England"
- A Local Authority Pension Fund

Allied Irish Bank / Allfirst

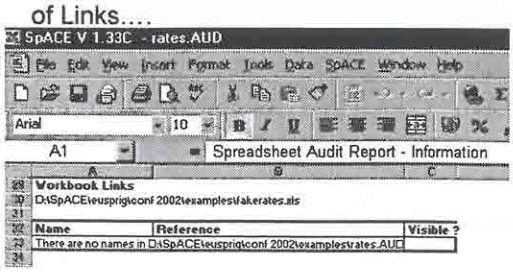
- Cause
  - Fraud involving (among other things) falsification of spreadsheets used for monitoring alleged perpetrator's work
    - Transaction entries / Contra – entries in Value At Risk spreadsheet
    - Exchange Rates ostensibly downloaded into spreadsheet from Reuters on-line feed
- Impact
  - Total US$691.2Million
  - False Bonus US$549,000 (not all paid)

Allied Irish Bank / Allfirst

- **Detection**
  - False entries – by accident when fraud already detected
  - False exchange rates – manual spreadsheet review by internal audit
- **Prevention**
  - Don't believe spreadsheets !
  - Monitoring / checking regularly
  - Source documents : spreadsheet check would have found false transactions
  - SpACE would have found the false rates

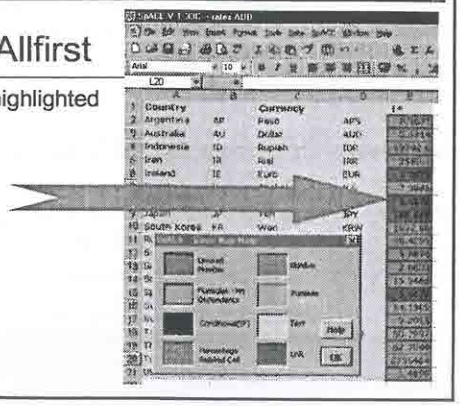
Allied Irish Bank / Allfirst

- SpACE would have found the Existence of Links....



AIB / Allfirst

- .And highlighted them



A School "Somewhere in England"

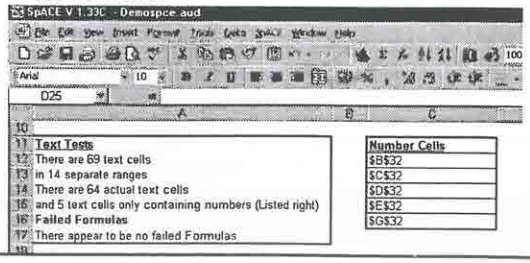
- Cause
  - Numbers as text…
  - …From ODBC Import
  - …Omitted by =Sum formulas
- Impact
  - £30,000 omitted from Budget so..
  - …Shortfall in school funding

A School "Somewhere in England"

- Detection
  - "That doesn't look right…."
  - After budget submitted to LEA
  - "… Oh #!x#!"
- Prevention
  - =IF(ISNUMBER(A1),"OK","ERROR") in strategic places
  - Build reasonableness checks in
  - SpACE would have found this in seconds

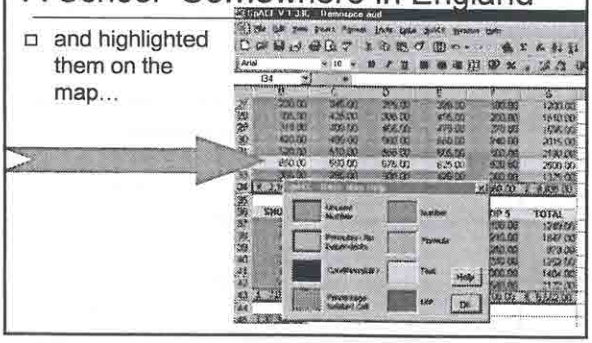
A School "Somewhere in England"

- SpACE Would Have Reported "Numbers as text"....



A School "Somewhere in England"

- and highlighted them on the map...



Local Authority Pension Fund

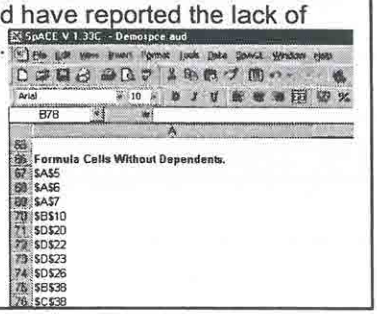
- Cause
  - Rows inserted, omitted from "bottom line"
- Impact
  - UK£ 4 million omitted from cash book...
  - Detected before irrevocable damage done

Local Authority Pension Fund

- Detection
  - SpACE Audit found formulas with no dependents
- Prevention
  - Better testing, and standards needed

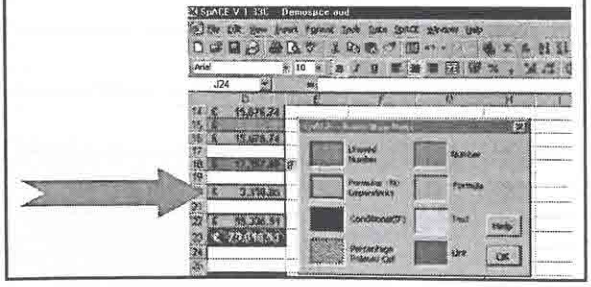
Local Authority Pension Fund

- SpACE would have reported the lack of dependents..



Local Authority Pension Fund

- ...And highlighted them on the map



Common Threads

- Lack of checking until it was too late
- Lack of control over development
- Lack of checks within the spreadsheets used
- Poor training of users..
- …In fact the usual suspects

Conclusion

- Lots more to do to get our point across
- Big potential market for education in Modelling with spreadsheets rather than making them work
- Big market for testing tools

Discussion / Questions

Sources

- AIB / Allfirst
  - Interim report posted www.aibplc.com
- School Budget
  - www.Accountingweb.co.uk
- Local Authority
  - Private communication with SpACE Customer

A TYPICAL MODEL AUDIT APPROACH

Spreadsheet Audit Methodologies in the City of London

Grenville J. Croll
EuSpRIG - European Spreadsheet Risks Interest Group, grenville@croll-management.freeserve.co.uk

Abstract: **Spreadsheet audit and review procedures are an essential part of almost all City of London financial transactions. Structured processes are used to discover errors in large financial spreadsheets underpinning major transactions of all types. Serious errors are routinely found and are fed back to model development teams generally under conditions of extreme time urgency. Corrected models form the essence of the completed transaction and firms undertaking model audit and review expose themselves to significant financial liability in the event of any remaining significant error. It is noteworthy that in the United Kingdom, the management of spreadsheet error is almost unheard of outside of the City of London despite the commercial ubiquity of the spreadsheet.**

Key words: Spreadsheet Model Risk Audit Review Error Integrity

1. INTRODUCTION

The salient characteristics of major financial transactions presently completed in the City of London and elsewhere are invariably modelled in large Microsoft Excel spreadsheets.

The financial magnitude of these transactions in Sterling is typically large - routinely hundreds of millions, often billions and occasionally much larger still. Clearly, with such significant value at stake, it is hardly surprising that the issue of spreadsheet error was recognised and dealt with at an early stage in the City of London.

The two original spreadsheet error detection packages, Cambridge Spreadsheet Analyst and Spreadsheet Auditor were certainly in use in the City of London during the mid eighties. Specialist businesses dedicated to spreadsheet based financial modelling, including the detection and correction of spreadsheet error were established by the early nineties. Specialist financial modelling teams with a model audit and review capability presently exist within the four large professional services firms, and some of the smaller firms, operating in the City.

The terms model audit and model review are used interchangeably. The former tends to be avoided by the larger firms due to liability issues and possible confusion with the corporate audit function. Given the exhaustive nature of the process and the financial liabilities of non-performance, the term model audit would, however, seem to be the most appropriate.

This article describes a spreadsheet model audit process typical of that presently used in the City of London.

2. MODEL AUDIT PROCESS

Spreadsheet models typically arrive for audit a few days prior to a transaction's financial completion. The audit process is one of the last procedures during a transaction and tends to get overlooked during the wider activities occurring at an earlier stage. This necessarily places constraints upon the order in which the various phases of model audit can take place.

After a brief familiarisation activity where the structure of the model in terms of sheets, linked sheets and associated databases is appraised, the model is partitioned and dispatched to one or more reviewers for detailed low level scrutiny. Model review often takes place in parallel, where different parts of the model are reviewed simultaneously. Model review is typically performed by graduates with or working for a formal accounting qualification.

Panko [1] has demonstrated that independent cell by cell inspection is almost the only process capable of systematically discovering errors. This is the approach adopted, often with the assistance of proprietary software tools. Model errors are fed back to the developer for correction using a simple error management system.

Microsoft Excel spreadsheets for review are invariably several megabytes with 10Mb not uncommon and 100Mb not unknown. Numbers of unique formulae for inspection are in the range 1,000-10,000 and upwards.

The time taken to review these models can range from twenty five hours to many hundreds, generating significant fee income for firms undertaking this work.

After the low level review has established the correctness of the detailed model formulae and other model parts, a high level review takes place. During the high level review, wider issues such as the correct handling of interest, tax and other financial and accounting issues is determined.

Once the model is correct, it can then be used to investigate sensitivities. That us to say, a few key model variables such as interest rates, cost and revenue assumptions are changed to review the characteristics of the transaction under various commercial scenarios. The performance of the model under these scenarios is checked and documented.

After numerous feedback iterations with the model developer have taken place to correct errors, and with financial closure invariably pressing, a report is issued to the client confirming that various previously agreed review procedures have been performed upon the model. Any unresolved issued are documented.

Financial transactions do not complete until the model audit has been completed. Any delay delays the transaction. Serious model problems have been known to cause transactions to collapse. The legal documentation supporting the transaction reflects the model and not necessarily vice versa. It is increasingly common for the model as audited and agreed at financial close to be the legally agreed tool for monitoring and controlling some key aspects of the deal post financial close. Covenants and restrictions regarding drawdown, repayments and distributions are often enforced via tests run using the original financial model, possibly over a twenty plus year loan life.

3. LOW LEVEL REVIEW

A variety of commercial and proprietary software tools exist to determine the likelihood and severity of error in client spreadsheets and the likely location of errors. Most review teams use a

variety and combination of tools such as OAK, Space, Spreadsheet Detective and Spreadsheet Professional (reviewed by Nixon and O'Hara in [2]) to increase the chance of error discovery.

Discoverable model characteristics such as formula length, ratio of original to repeated cells, numbers of cell precedents and dependents and the locality and non-locality of cell linkages can be used to infer information about the relative ease or difficulty and time to review a given model.

High level maps of a spreadsheet where the contents of each cell are denoted by a single character such a L for label cell and F for formula cell etc. are commonly used to help ensure that every single cell is examined.

Considerable attention is devoted to ensuring the correctness of ranges and range names, including the following of range naming conventions. Typical problems include overlapping ranges, empty ranges and ranges which do not cover the intended set of cells. Establishing the correctness of ranges early in the low level review process assists with later work.

During the process of examining the formulae in a spreadsheet a large number of issues are considered. These issues can include: ensuring all references to other cells are correct; arguments of functions (such as IF, INDEX, SUM, VLOOKUP etc) are correct; correct use of the ROUND functions; absence of technical errors such as #ERR, #REF etc; absence of circularity; absence of embedded constants; consistency of units of weight and measure; absolute and relative cell addresses and the correct accounting sign of the intermediate and final results.

The documentation trail for a model audit can include a printout of a high level map for the entire spreadsheet with every examined cell ticked and each sheet numbered, signed and indexed.

Most financial spreadsheets contain additional Visual Basic and Macro code which must also be checked. This is done by printing out, inspecting, running and testing all executable entities. This part of the process of reviewing a spreadsheet is directly comparable with the code review phase of traditional software engineering. Grossman [3] deals more fully with the comparisons between software and spreadsheet engineering.

4. HIGH LEVEL REVIEW

Having established the integrity of the low level formulae and code within a model, a high level review can take place. The high level review takes place to check aspects of the overall integrity of the model that may have been missed by the low level review.

A high level review will generally address the consistency of the model with any supporting documentation. This part of the review will also include checks to ensure that finance and accounting issues have been dealt with correctly. That is to say that the model follows Generally Accepted Accounting Practices (GAAP) relevant to the jurisdiction within which the transaction takes place.

Depending upon which firm is performing the model audit, the high level review may or may not address commercial issues relevant to the transaction. Often it is too late to address issues such as the viability or commercial logic of the transaction and its terms. It is often the case, however, that the true nature of the transaction only becomes visible once a correct model is in place.

Typical high level checks for a financial model cover issues such as: ensuring the balance sheet balances; checking whether retained earnings flow from the profit and loss account to the balance sheet; ensuring debt is amortised correctly; ensuring fixed assets do not depreciate below zero;

checking whether revenues and costs reflect production; ensuring tax and deferred tax is handled correctly and so forth.

High level checks on the model and its documentation will certainly include a detailed review of the term sheet of any debt or other funding to ensure that the precise details of its provision are reflected in the model. The funding provisions can be very detailed, particularly where financial reserves are accumulated to ensure debt is serviced correctly. Other documentation checks can include detailed matching of model constants such as revenue and cost estimates to the original documents containing them and vice versa to ensure that the model contains the required detail.

Any changes to the model at the high level review stage cause a re-review of the model at low-level. Comparison software is often used to assist in locating model changes between versions.

5. SENSITIVITIES

By now, the model is largely correct. That is to say, if a specification existed (more often than not, they do not), the model would largely perform to that specification.

Running sensitivities involves running the model with a handful of key variables changed either singly or in combination. Key variables include revenue & costs - both high and low, start date, interest rates, discount rates and debt/equity ratios. For a large model, running sensitivities can be time consuming with model re-calculate times of several hours not being unknown. For a well designed and constructed model, the sensitivities will run without error, yielding financial information essential to the completion of the transaction. Errors do occur at this late stage, requiring rapid repair and re-review prior to re-execution of all the sensitivities.

The results of each sensitivity are methodically reviewed and documented. A version of the model containing the changed variables will be saved.

6. FINAL REVIEW & REPORT

The concluding part of a model audit is a drawing together of all the work that has been performed upon the model. A package of documentation will be assembled which can be used (in court if necessary) to prove that every check and test that was required to be performed was performed.

Any final queries regarding model performance will be cleared if possible. Any areas of concern that remain will be clearly documented.

The wording of the final report to the client will vary according to which firm is performing the work, what type of work was performed and who the client is. Wording can vary from simply stating that various agreed upon procedures have been performed through to statements that the model is free of material error.

Care is taken in final reports to identify exactly which model was the subject of the audit. Typical identification data will include basic information such as file name, date, time and size.

The limit to the contractual liability of the firm performing the model audit work is an important issue. A liability figure is always agreed prior to any work taking place and often re-stated in the final report. In some cases liability is, or has to be, unlimited, placing onerous responsibilities upon firms and individuals that perform this work on large transactions.

7. CONCLUSION

Model audit is a detailed, time consuming and essential prerequisite for the consummation of financial transactions in the City of London. Correctness of spreadsheet models ensures that the parties to a transaction can rely upon the integrity of the financial information presented.

The methodologies used for ensuring the correctness of spreadsheet models are generally straightforward and have parallels in more widely studied aspects of software engineering, where code review is the norm.

Analysis of the client bases of firms performing model audit suggests that model audit is largely confined to financial work performed in the main financial centres. Given the ubiquity of the spreadsheet and spreadsheet error, the integrity and reliability of spreadsheets outside of this narrow field is questionable.

8. ACKNOWLEDGEMENTS

The author thanks David Chadwick, Ray Butler and Penny Lynch for their constructive comments upon a draft of this paper.

9. REFERENCES

[1] Panko R. *Spreadsheet Errors: What We Know. What We Think We Can Do,* Proceedings of the First Symposium of the European Spreadsheet Risks Interest Group, Greenwich, UK, 2000.

[2] Nixon D, O'Hara M, *Spreadsheet Auditing Software,* Proceedings of the Second Symposium of the European Spreadsheet Risks Interest Group, Greenwich, UK, Amsterdam, 2001

[3] Grossman T.A., *Spreadsheet Engineering: A Research Framework*, Proceedings of the Third Symposium of the European Spreadsheet Risks Interest Group, Greenwich, UK, Cardiff, 2002