Managing Spreadsheets in the light of Sarbanes-Oxley
European Spreadsheet Risks Interest Group

EuSpRIG 2005 Conference

Managing Spreadsheets in the light of Sarbanes-Oxley

Editor:
David Ward

Conference Hosted By:
David Chadwick,
University of Greenwich, United Kingdom

Conference Sponsored By:
Smart Tech Consulting Services Ltd
&
Frontline Systems UK Ltd
Information Systems Audit & Control Association
(Northern England Chapter)
KPMG LLP
PriceWaterhouseCoopers LLP
The University of Greenwich
University of Wales Institute Cardiff

July 2005
The University of Greenwich, London
Proceedings of EuSpRIG 2005 Conference
Managing Spreadsheets in the light of Sarbanes-Oxley

First published 2005
Printed in United Kingdom by KPMG LLP

Held at University of Greenwich; 7 to 8 July 2005

Copyright: European Spreadsheet Risks Interest Group
www.eusprig.org

ISBN :

All rights reserved. Apart from fair dealing for the purposes of research or private study, or criticism or review, as permitted under the Copyright, Designs and Patents Act 1988, no part of this publication may be reproduced, stored in a retrieval system, or transmitted in any form by any means, electronic, mechanical, photocopied, recording or otherwise, without prior permission of the Publisher who is the Committee of the European Spreadsheet Risks Interest Group (www.eusprig.org).

Conference Organised by
EuSpRIG Committee

The use of registered names, logos, trademarks etc. in this publication does not imply, even in the absence of a specific statement, that such names are exempt from the relevant laws and regulations and therefore free for general use.
No responsibility is assumed by the Publisher for any injury and/or damage to persons or property as a matter of products liability, negligence or otherwise, or from any use or operation of any methods, products, instructions or ideas contained in the materials herein.

Cover design by David Ward of KPMG with photographs donated by University of Greenwich, United Kingdom
EuSpRIG 2005 Conference
"Managing Spreadsheets in the light of Sarbanes-Oxley"

Organising Committee
Chair: Patrick Cleary, University of Wales Institute Cardiff, UK
David Chadwick, University of Greenwich, UK
Gary Owen, University of Greenwich, UK
Grenville Croll of Frontline Systems UK
Graham MacDonald, HM Customs & Excise, UK

Programme Committee
Chair: Patrick Cleary, University of Wales Institute Cardiff, UK
Simon Thorne, University of Wales Institute Cardiff, UK
Patrick O’Beirne, Systems Modelling Ltd, Ireland
Ray Butler, Highways Agency, UK
Markus Clermont, University of Limerick, Ireland
David Colver, Operis, UK
Prof. Roland Mittermeir, University of Klagenfurt, Austria
Jocelyn Paine, Bristol University, UK
Prof Ray Panko, University of Hawai’i
Barry Pettifor, PriceWaterhouseCoopers LLP, UK
Leon Strous, De Nederlandsche Bank, Netherlands
David Ward, KPMG LLP, UK

EuSpRIG Management Committee
EuSpRIG Chair: Patrick O’Beirne
EuSpRIG Secretary and Chair of Conference Programme Committee: Patrick Cleary
EuSpRIG Membership Secretary: Grenville Croll
EuSpRIG Treasurer: Graham MacDonald
Member: Ray Butler
Member and past Chairman: David Chadwick:
  Member: Markus Clermont
  Member: Prof. Roland Mittermeir
  Member: Jocelyn Paine
  Member: Barry Pettifor
  Member: Leon Strous
  Member: David Ward
PREFACE

Welcome

We are delighted to welcome you to the biggest and most successful Eusprig conference yet! Thank you for coming, your interest is the reason for our conference.

Your presence is important to us and to the wider community of risk management professionals. You can bring back what you will learn, raise the awareness of spreadsheet risk, improve practices, and assist your organisation with compliance.

Our development

In six years, the European Spreadsheet Risks Interest Group (EuSpRIG) has grown in strength from a European to a worldwide network of academics and practitioners. Our contributors this year are from Austria, Canada, Ireland, Israel, Netherlands, the UK, and USA. In previous years we have had papers from Australia, Japan, New Zealand, and Slovenia.

Our mission is to increase the awareness of spreadsheet risk and promote methods of assessing risk, detecting errors, and improving the productivity and quality of spreadsheet development.

The most significant step forward in the past year has been the culmination of our joint work with the European Computer Driving Licence Foundation (ECDLF) in the drawing up of a syllabus which will form the foundation of a new end user certification in Spreadsheet Check+Control. This will be welcomed by all stakeholders who are concerned about exposure to risk from uncontrolled spreadsheet development.

Standards

Our review process involves three referees who provide feedback comments to authors. We apply strict standards of references and evidence to the academic papers, and appropriate standards of relevance and usefulness to the submissions from practitioners. This year, we received more papers than we could cope with within our now established format of a day and a half. We were sorry to have to disappoint the authors of some good work, but the competition was strong.

Thanks

We are very thankful to the speakers, whose expertise, experience, and contribution is deeply appreciated and will enrich those who hear it. We were very fortunate this year in securing our keynote speakers, Professor Ray Panko of the University of Hawai‘i,
Dean Buckner of the Financial Services Authority, and Derek Wimmer of Wimmer Systems.

We are grateful to the sponsors who have supported us financially and professionally: our prime sponsor is SmartTech Consultancy Services Ltd (UK), along with Frontline Systems UK, Information Systems Audit and Control Association (Northern Chapter), KPMG UK, PriceWaterhouseCoopers, the University of Greenwich, and the University of Wales in Cardiff. These organisations have contributed a great amount of expertise in the organising of this conference, the publicity, the proceedings, and much more work done in the background. We deeply appreciate their unstinted contribution.

Special thanks are due to the committee who have given voluntarily of their time, expertise, and organisational resources, to bring this conference to its present success:

- Pat Cleary, who took on the roles both of secretary and Programme Committee chair and general factotum;
- Graham MacDonald, our treasurer who set up our euro processing facilities this year and keeps a keen eye on correct books;
- David Ward of KPMG for preparing the proceedings;
- Barry Pettifor of PwC who along with David Ward provided very valuable sponsorship of our teleconferences;
- Grenville Croll of Frontline Systems UK who manages our membership database and publicity, and performed the very important task of processing credit card payments for the conference;
- Ray Butler for the continuing underwriting by ISACA, and his own sage advice as a past chair;
- David Chadwick, our 2003 Chair, a founder member, and our member in Greenwich University;
- and all members of the EuSpRIG committee who gave of their time and thought in our monthly teleconference calls: Markus Clermont, David Colver, Roland Mittermeir, Jocelyn Paine, Leon Strous, and Simon Thorne.

Enjoy the conference, contribute to the discussions, benefit from the networking, and keep in touch!

Patrick O’Beirne, Chairman 2005, EuSpRIG

Greenwich, July 2005
# Contents

1 Regulatory Update  
Dean Buckner; UK Financial Services Authority  
1

2 Spreadsheet Management & Remediation Framework  
Barry Pettifor; PriceWaterhouseCoopers LLP  
7

3 Sarbanes–Oxley: What About All the Spreadsheets?  
Raymond R. Panko & Nicholas Ordway  
15

4 Spreadsheet Risk and Corporate Responsibility – are UK/EU Directors Liable?  
Pat Cleary, Lynne Norris-Jones, Mukul Madahar  
62

5 Protecting Spreadsheets against Fraud  
Roland T. Mittermeir, Markus Clermont, Karin Hodnigg  
69

6 The Importance and Criticality of Spreadsheets in the City of London  
Grenville J. Croll  
82

7 Regulation and the Integrity of Spreadsheets in the Information Supply Chain  
Ralph Baxter  
95

8 Qtier-Rapor: Spreadsheets in Compliance with the Sarbanes-Oxley Act  
Keith Bishop  
103

9 Breaking Out of the Cell: On The Benefits of a New Spreadsheet User-Interaction Paradigm  
Ziv Hellman  
113

10 Controlling the Information Flow in Spreadsheets  
Vipin Samar, Sangeeta Patni  
125
<table>
<thead>
<tr>
<th>Chapter</th>
<th>Title</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>11</td>
<td>Developing an Auditing Protocol for Spreadsheet Models</td>
<td>135</td>
</tr>
<tr>
<td></td>
<td>Stephen G. Powell, Kenneth R. Baker, Barry Lawson</td>
<td></td>
</tr>
<tr>
<td>12</td>
<td>The use of spreadsheets to enable Sarbanes-Oxley 404 compliance.</td>
<td>149</td>
</tr>
<tr>
<td></td>
<td>Adrian Carter</td>
<td></td>
</tr>
<tr>
<td>13</td>
<td>Why, How and When Spreadsheet Tests Should be Used</td>
<td>155</td>
</tr>
<tr>
<td></td>
<td>J C Nash and Jody Goldberg</td>
<td></td>
</tr>
<tr>
<td>14</td>
<td>Exploring Human Factors in Spreadsheet Development</td>
<td>161</td>
</tr>
<tr>
<td></td>
<td>S. Thorne, D. Ball</td>
<td></td>
</tr>
<tr>
<td>15</td>
<td>Excelsior: bringing the benefits of modularisation to Excel</td>
<td>173</td>
</tr>
<tr>
<td></td>
<td>Jocelyn Paine</td>
<td></td>
</tr>
<tr>
<td>16</td>
<td>A Revised Classification of Spreadsheet Errors</td>
<td>185</td>
</tr>
<tr>
<td></td>
<td>Kamalasen Rajalingham</td>
<td></td>
</tr>
<tr>
<td>17</td>
<td>Comparison of Spreadsheets with other development tools</td>
<td>201</td>
</tr>
<tr>
<td></td>
<td>(limitations, solutions, workarounds and alternatives)</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Simon Murphy</td>
<td></td>
</tr>
<tr>
<td>18</td>
<td>Remediation Services for Excel, a comprehensive approach</td>
<td>209</td>
</tr>
<tr>
<td></td>
<td>Freek Varossieu, Cheryl O'Neil, Jan Prauis</td>
<td></td>
</tr>
<tr>
<td>19</td>
<td>Archiving: The Overlooked Spreadsheet Risk</td>
<td>213</td>
</tr>
<tr>
<td></td>
<td>Dr. Victoria Lemieux</td>
<td></td>
</tr>
</tbody>
</table>
1 Regulatory Update

Dean Buckner; UK Financial Services Authority

Regulatory Update

Dean Buckner
UK Financial Services Authority

July 2005

Summary

• In 2003 I made keynote speech to Eusprig
  – Highlighting areas where FSA concerned about spreadsheet use in financial industry
• Today I shall review progress made in these areas
  – As always, there is good news and bad
Issues in 2003

- Change of mindset (senior mgt, IT)
- User training
- No "good practice"
- Accreditation
- Audit awareness
- Data standards
- The "M" problem

Management mindset

- We would like firms to have an IT strategy on spreadsheets
  - More than just "aim to replace them!"
- Some good news
  - IT directors increasingly admit need for strategy
  - One bank now explicitly recognise this (Case study 1)
- Bad news
  - Hard to see "big picture"
User training

- Most problems the result of poor use of EUC solutions
  - Falls under “stupid practice”
  - (Case study II)
- Good training the obvious solution
- But no IT budget!
  - This would mean explicit acceptance of supposedly “tactical” solutions
  - Connects with IT strategy issue

Good practice

- Good news
  - EUC policy now becoming a standard (in banks at least)
- Bad news
  - Policy tends to be very high level
  - Eusprig has no view on good practice
  - FSA’s position is that this is industry issue
Accreditation

- No good news
  - Accreditation seen as burdensome and risky
  - And difficult (implies generally accepted view on good practice, for a start)

Audit

- Very good news
  - We are seeing more mention of spreadsheets in audit reports
  - Doesn't mean spreadsheet use increasing!
  - Does mean that auditors now recognise spreadsheets exist!
  - And EUC part of audit plan
- Thanks to Eusprig
Data

• **Data processing is the biggest problem**
  - Mostly done on spreadsheets, via ad-hoc and "hacky" downloads
  - Increasing use of ACCESS in bad ways
  - No "vested interest" in good data (even in Eusprig community)

• **Basel has helped a bit**
  - Concept of "data ownership"

"M" problem

• **Some good news**
  - Dialogue opened up with banks and FSA
  - Apparent willingness

• **But obvious difficulties**
  - Which problems are "M" specific?
  - Or is it just bad use of a good tool?
  - Some issues: code fragmentation, poor help, poor data transfer (try moving UK date formats from ACCESS to EXCEL), 256 column problem, ACCESS security &c &c
Questions & Comments
2 Spreadsheet Management & Remediation Framework

Barry Pettifor: PriceWaterhouseCoopers LLP

Spreadsheet Security and Control
How Do We Think About Spreadsheets - Inside Out.

Our Clients Come To Us With Several Business Issues...

- Spreadsheets used in critical finance functions are a regulatory liability
- Reports are not timely and don't facilitate business decision making
- Quality of data in the reports is not completely reliable
- Numerous spreadsheets are interlinked without appropriate controls
- Manual tasks result in costly and error prone report production cycle
- Data used for reports is stored in multiple places and is difficult to aggregate
- Ad-hoc analysis is time consuming due to the lengthy setup-time required to get all the required data in one place
- Data analysts are performing "Data Grunt" activities
- Key spreadsheets are not documented and this results in significant key-person risk
- Complex spreadsheet models used to support M&A deals may not be accurate
- Existing spreadsheets are "black boxes" and are not understood
- Short term solution needed for time constrained reporting
Spreadsheets are here to stay... Every production system WILL have spreadsheets filling its requirements gap

- Production systems capture requirements at a specific point-in-time
- User requirements are dynamic
- Users create Spreadsheets to fill the Requirements Gap

* A Production System Becomes Obsolete As Soon As It Goes Live!
"Spreadsheet Management and Remediation" (SMR) is a unified approach towards creating and improving controls, efficiency and reliability of spreadsheet based processes.

1. Designing and improving controls (SOX, regulatory)
2. Making a spreadsheet process more efficient
3. Building/validating reliable spreadsheet models

Spot Improvement Opportunities – Know Appropriate Uses of Spreadsheets

One way spreadsheets are mostly "OK", use data, analyze and print (use excel to analyze data)
Two way spreadsheets are evil (data comes in, is processed and then results are used elsewhere)
What the businesses would like to do but can't due to data unavailable

Analytics spreadsheets (used to identify data and make decisions)
Interlinked Spreadsheets
Manual entry

OK
Not OK
Bad
Very Bad

PricewaterhouseCoopers
Improving the Spreadsheet Environment Using a Standards based approach (Self Discovery)

- **Cost**
- **Timeliness**
- **Retain flexibility**
- **Reduced Risk**

**Standardize**

1. Analytics spreadsheets (used to illustrate data and make decisions)
2. Interlinked Spreadsheets
3. Database Driven Spreadsheets (bidirectional)

**Automation**

**Good → Better**

**Not Good → Better**

**Bad → Better**

**Improved Controls**

Better Data Quality

---

Typical Permanent Solutions for Spreadsheet Environment

1. **Governance**
   - Identify a "Spreadsheet Steward" (IT + Business)
   - Incentives to improve controls and efficiency
2. **Standards**
   - Database, spreadsheet policies
3. **People**
   - Excel/Access Training
   - Awareness of production tools for reporting & analytics
4. **Process**
   - Redesign and automate
5. **Technology/Tools**
   - Exception based control
   - Data-sharing through database

---
Why Do We Need To Think About Controls, Efficiency and Modeling Drivers In A Unified Manner?

- Because approaching any of these improvement drivers in isolation may negatively impact the others:
  - Focusing only on controls improvements may result in a highly inefficient models
  - Focusing on efficiency in isolation may result in poor controls and inflexible and unreliable model
  - Etc.
Checklist for Improving a Spreadsheet Environment

✓ 1- Understand the big picture
   ✓ How/why does data move (data flow diagrams)
   ✓ Where are production systems and spreadsheets
✓ 2- Inventory and prioritize spreadsheets
   ✓ Which ones are critical are require "attention" (controls/efficiency/reliability)
✓ 3- Identify Improvement Opportunities across four reporting steps
   ✓ Collect & Prepare
   ✓ Validate & Store
   ✓ Aggregate & Analyze
   ✓ Publish & View
✓ 4- Translate improvement opportunities to specific changes in
   ✓ People
   ✓ Process
   ✓ Tools/Technologies
   ✓ Governance
   ✓ Standards
✓ 5- Prioritize improvement Opportunities (cost/benefit analysis)
✓ 6- Structure a tactical project for a specific process which would take 4-6 weeks to improve
✓ 7- Design, Implement, test, deploy
✓ 8- Take the next set of improvement opportunities (go back to step 6 and identify additional set of opportunities to tackle)

EuSpRIG 2005 will have been a success if....

• We can raise awareness amongst business leaders of the scale and nature of the issues;
• We can get people focused on when it's smart to use Spreadsheets, and when there are better ways;
• We can define and share useful ideas to:
  - Achieve efficiency
  - Make controls more effective
  - Help people get more out of their analysis work.
• We have FUN!
Sarbanes-Oxley: What About all the Spreadsheets?

Controlling for Errors and Fraud in Financial Reporting

Raymond R. Panko & Nicholas Ordway

Raymond R. Panko
University of Hawai'i
panko@hawaii.edu

Nicholas Ordway
University of Hawai'i
olgierd@hawaii.edu

Abstract

The Sarbanes-Oxley Act of 2002 has finally forced corporations to examine the validity of their spreadsheets. They are beginning to understand the spreadsheet error literature, including what it tells them about the need for comprehensive spreadsheet testing. However, controlling for fraud will require a completely new set of capabilities, and a great deal of new research will be needed to develop fraud control capabilities. This paper discusses the riskiness of spreadsheets, which can now be quantified to a considerable degree. It then discusses how to use control frameworks to reduce the dangers created by spreadsheets. It focuses especially on testing, which appears to be the most crucial element in spreadsheet controls.

Keywords

Cell error rate (CER). CobiT. controls. control deficiency. control framework. COSO. end-user computing (EUC). error. error rate floor. fraud. 17799. ITIL. material. spreadsheet. testing.

Figures

Figures are printed at the end of the paper.

Introduction

Sarbanes-Oxley

In 2002, after financial reporting frauds at Enron and other major companies, the U.S. Congress passed the Sarbanes-Oxley Act. This act covers many things, including the retention of documents. However, the focus of the Act, which is often called SOX, is Section 404. This
section requires chief corporate officers to assess whether their financial reporting systems are effective. Furthermore, it assumes an independent external auditor must assess the officers' assessment.

To implement SOX, Congress created the Public Company Accounting Oversight Board (PCAOB) to create auditing standards. PCAOB's main guidance on 404 assessments of control attestations has been Auditing Standard No. 2, An Audit of Internal Control Over Financial Reporting Performed in Conjunction with an Audit of Financial Statements.

The focus of SOX and of Auditing Standard 2 is the creation of effective controls. Figure 1 illustrates that controls are ways to help a corporation achieve its goals, such as producing accurate financial reports. Controls cannot guarantee that the goals will be met, but they reduce the risk that these goals will not be met. In this context, effectively controlled financial reporting processes give reasonable assurance that the company will meet the goal of producing accurate financial reports.

Effectively controlled financial reporting processes give reasonable assurance that the company will meet the goal of producing accurate financial reports.

Figure 1: Controls

According to Standard 2, an internal control deficiency exists when the design or operation of a control does not allow for the timely prevention or detection of misstatements. The standard defines two types of deficiencies:

- In a significant deficiency, there is more than a remote likelihood that the financial statements will be impacted in a manner that is consequential but not material.

- In a material deficiency, there is "a significant deficiency, or combination of significant deficiencies, that results in more than a remote likelihood that a material misstatement of the annual or interim financial statements will not be prevented or detected." Vorhies [2005] indicates that a 5% error in revenues is the usual threshold for labeling a deficiency as material because a smaller difference is not likely to sway a reasonable investor.

This distinction between significant and material internal control deficiencies is important because if management finds even a single material deficiency, it is not allowed to assess its internal controls as effective.

The Use of Spreadsheets in Financial Reporting

Auditing Standard No. 2 clarifies that controls must involve all forms of information technology (IT) used in financial reporting. One particular IT concern for corporations is the use of spreadsheets in financial reporting. There have long been indications that many spreadsheets are large [Cale, 1994; Cragg & King, 1993; Floyd, Walls, & Marr, 1995; Hall, 1996], complex [Hall, 1996], and very important to their firm [Chan & Storey, 1996; Gable, Yap & Eng, 1991; Hall, 1996:]. When Comshare, Inc. surveyed 700 finance and
budgeting professionals in the mid-1990s, it found that spreadsheets were already dominating budgeting [Modem Office Technology, 1994]. In turn, when CPS [2001] did a survey of strategic planning practices in major U.S. corporation, 72% of respondents relied on spreadsheets exclusively to do strategic planning.

Although some people may doubt that companies use spreadsheets in critical financial reporting operations, the widespread use of spreadsheets is well documented.

- Financial intelligence firm CODA reports that 95% of U.S. firms use spreadsheets for financial reporting according to its experience (www.coda.com).

- In 2004, RevenueRecognition.com [2004] (now Softtrax) had the International Data Corporation interview 118 business leaders. IDC found that 85% were using spreadsheets in financial reporting and forecasting.

- CFO.com [Durfee 2004] interviewed 168 finance executives in 2004. The interviews asked about information technology use in the finance department. Out of 14 technologies discussed, only two were widely used—spreadsheets and basic budgeting and planning systems. Every subject said that his or her department used spreadsheets.

- In 2003, the Hacket Group (www.thehacketgroup.com) surveyed mid-size companies. It found that 47% of companies use stand-alone spreadsheets for planning and budgeting.

- In Europe, A.R.C. Morgan interviewed 376 individuals responsible for overseeing SOX compliance in multinationals that do business in the United States [TMCnet.com, 2004]. These respondents came from 21 different countries. More than 80% of the respondents said that their firms use spreadsheets for both managing the control environment and financial reporting.

- In a webcast for Deloitte on May 22, 2005, the first author was able to ask a series of questions of the audience. The average response size was just over 800 financial professionals and officers in corporations. One question specifically asked, "Does your firm use spreadsheets of material importance in financial reporting?" Of the respondents, 87.7% answered in affirmative, while 7.1% said, "No." (Another 5.2% chose "Not Applicable.")

Furthermore, when companies use spreadsheets for financial reporting, they often use many. One firm used more than 200 spreadsheets in its financial planning process.

Today, companies are widely confused over what to do about spreadsheet errors. Obviously, if financial reporting spreadsheets contain a significant number of errors and a reasonable amount of testing has not been done, it is difficult to say that the reporting process is well controlled.
Most firms will have to implement Section 404 for their Year-End 2005 financial reports. Most firms and external auditing companies are likely to issue vague statements about their spreadsheet controls or are likely to ignore spreadsheets entirely. However, as we will see below, these assessments are likely to be wrong.

Lack of Controls

One concern with spreadsheets is that they are rarely well controlled [Davies & Ikin, 1987; Cragg & King, 1993; Fernandez, 2002; Floyd, Walls, & Marr, 1995; Gosling, 2003; Hall, 1996; Hendry & Green, 1994; Nardi & Miller, 1991; Nardi, 1993; Schultheis & Sumner, 1994]. This is not surprising because few organizations have serious control policies—or indeed any policies at all—for spreadsheet development [Cale, 1994; Fernandez, 2002; Floyd, Walls, & Marr, 1995; Galletta & Hufnagel, 1992; Hall, 1996; Speier & Brown, 1996].

A specific concern is testing. Although there has long been evidence that spreadsheet error, at least, is widespread, organizations rarely mandate that spreadsheets and other end user applications be tested after development [Cale, 1994; Cragg & King, 1993; Floyd, Walls, & Marr, 1995; Galletta & Hufnagel, 1992; Gosling, 2003; Hall, 1996; Speier & Brown, 1996]. Also, individual developers rarely engage in systematic testing on their own spreadsheets after development [Cragg & King, 1993; Davies & Ikin, 1987; Hall, 1996; Schultheis & Sumner, 1994].

As noted earlier, the first author was able to ask questions of corporate financial professionals and officers in a webcast. Figure 2 shows respondent answers when they were asked, "For spreadsheets of material importance used in financial reporting, what percentage does your company test?" By and large, this figure shows that few spreadsheets are tested, even if they are of material importance. However, it shows that 17% of the respondents said that their firm tests more than 25% of their material financial spreadsheets, and 16% said that their firm tests nearly all. This makes it appear that a sizeable minority of companies do test their spreadsheets. However, as we will see later, what most respondents call testing is "looking over the spreadsheet," not comprehensive cell-by-cell testing.

Figure 2: Testing for Material Financial Spreadsheets

This lack of comprehensive testing may exist because developers tend to be overconfident of the accuracy of their untested spreadsheets. Certainly, widespread overconfidence, often in the face of widespread errors, has been seen repeatedly in spreadsheet research [Brown & Gould, 1987; Davies & Ikin, 1987; Floyd, Walls, & Marr, 1995; Panko & Halverson, 1997; Panko & Featherman, 2005].

In a vicious cycle, organizations that do not test their spreadsheets get no feedback on real error rates and so do not realize the ubiquity of spreadsheet errors, and, therefore, see no need for testing. Rasmussen [ 1974] has noted that people use stopping rules to decide when to stop doing activities such as testing. If people are overconfident they are likely to stop too early. Consequently, if firms use spreadsheets to make decisions but do not test their spreadsheets, they may not realize how many errors there are in their spreadsheets.
One might argue that the real world would provide painful feedback if a spreadsheet were incorrect. For some situations, such as budgeting, errors would have to be small in order to pass undetected. Unfortunately, in this case, even small percentage errors can be very damaging. Hicks [1995] found that a relatively small percentage error in the capital budgeting spreadsheet he examined would have produced an error of several hundred million dollars. At the other extreme, when a new situation is modeled, such as the purchase of another company, even large errors in the spreadsheet might not be obvious. If a promising corporate purchase goes bad, furthermore, it would be easy to dismiss the problem as being due to unforeseen factors even if the real problem is a spreadsheet error. Without testing, real-world feedback may not be very effective.

Overall, spreadsheet errors are widespread and there are strong prospects for fraud. This situation is at odds with the looseness with which most organizations treat their spreadsheets.

The Prevalence of Spreadsheet Errors

One problem with spreadsheets is the specter of errors. If even a single spreadsheet has more than a remote likelihood of creating a material financial error through having errors, the organization will not be able to assess its entire financial reporting system as effective. In addition, if a company uses a large number of spreadsheets in financial reporting—as most firms do—then a collection of spreadsheets that individually only have significant potential deficiencies because of errors may constitute a material internal control deficiency.

Are errors common in spreadsheets? The answer is that errors are extremely common. The most convincing data on this come from audits of real-world operational spreadsheets. Figure 3, which presents data from several audit studies, shows convincingly that spreadsheet errors are anything but rare.

**Figure 3: Audits of Real-World Spreadsheets**

- First, these audits found errors in nearly all (94%) of the spreadsheets they audited. This percentage would have been even higher, but several of the studies only reported serious errors. In other words, we can expect nearly all spreadsheets to contain errors.

- Second, these audits found many errors in the spreadsheets they audited.

- Specifically, studies that measured errors on a per-formula basis [Butler, 2000; Clermont, Hanin, & Mittermeier, 2002; Hicks, 1995; Lawrence & Lee, 2004; Lukasik, 1998] found errors in 5.2% of the formulas in these spreadsheets. Most large spreadsheets have thousands of formula cells, so most large spreadsheets probably have dozens or even hundreds of errors.

If this cell error rate seems excessive, it should not. There has been a great deal of research on human error [Panko, 2005a], and for tasks of comparable complexity, such as writing computer program statements, similar error rates are also seen. Panko (2005a) has summarized results from a number of studies that measured fault rates in real-world software. Of particular value are four
large studies [Madachy, 1996; O'Neill, 1994; Ebenau & Strauss, 1994; Weller 1993]. In these studies, average fault rates ranged from 1.5% to 2.6%. Note that this is very close to the cell error rates seen in Figure 3 for spreadsheet code inspection. Zage and Zage [1993] and Grady [1992] both found that fault rates depend on program difficulty. In both studies, fault rates were at least twice as high for difficult programs as for simple programs.

Humans appear to have an error rate floor that exists despite working very carefully. Everyone has a similar error rate floor, and working more carefully can decrease this error rate only modestly. Research has shown that the same human cognitive processes that allow us to respond to the world correctly most of the time have unavoidable trade-offs that create errors a few percent of the time [Reason, 1990]. In most human cognitive activities, such small error rates are only minor nuisances, if anyone notices them at all. However, when dozens of formula cells are on a chain to calculate a bottom-line financial value, the probability of error in the bottom-line value becomes unacceptable.

How many spreadsheet errors are significant or material? (As noted earlier, a 5% error in an important bottom-line value in a key financial variable would probably be considered a material error [Vorhies, 2005].) Unfortunately, we lack hard data on this critical issue. However, when Panko [2005b] interviewed the two spreadsheet auditing principals, both independently gave data suggesting that about 5% of all spreadsheets contain what one of the interviewees called "show stopper" errors. In addition, the Coopers and Lybrand [1997] study shown in Figure 3 did not report an error unless there was at least a 5% error in a bottom line value. The study found such errors in 91% of all spreadsheets. KPMG found a similar error rate and only reported spreadsheets to be incorrect if they contained errors that would make a difference to decision makers.

We also have data from software inspections, which sometimes classify errors as major or minor. Although definitions about what constitutes a major error differ, all software audit studies that have used the major/minor distinction found that major errors are very common. Bush [1994] and Jones [1996] both reported that a quarter of the errors in the inspections they examined were major errors. O'Neill [1994] found only 13% of errors to be major, while Schumeyer [1999] found that 42% of all errors were major. In addition, Ebenau and Strauss [1994] and Weller [1993] found major errors in 1.4% to 2.8% of the lines of code examined but did not report major errors as a percentage of total errors. Given this data from software inspections, it certainly would be risky to assume that nearly all spreadsheet errors will be minor.

**The Prospect of Spreadsheet Fraud**

Although there has been a great deal of research on spreadsheet error, there has not been formal research on spreadsheet fraud. Legally, a fraud exists when one person knowingly lies or conceals information whose nondisclosure would make other statements misleading, in order to get the victim to act in a way contrary to the victim's interests. Note that two elements are needed for there to be fraud: deception and harm.

There has long been concern that spreadsheet developers will use optimistic (or pessimistic) assumptions to make their plans and actions look better [Levy, 1984]. However, few
people would consider this "puffery" to be fraud because people who read spreadsheets should be somewhat skeptical consumers of spreadsheet results. However, if people to whom spreadsheets results are submitted are not likely to be aware of the degree of risk, this is at least an ethical violation [Newell, 1995]. In addition, when the damage due to deception increases in intensity, then spreadsheet misanalysis rises to the level of fraud.

H. M. Customs and Excise

Spreadsheet fraud is not just a theoretical concern. In England, the H. M. Customs and Excise collects certain types of taxes. When spreadsheets became prevalent in tax submissions, the agency began to audit submitted spreadsheets and found that many had substantial errors. In the late 1990s, the agency developed a program to automate many aspects of this auditing process [Butler, 2000]. This was SpACE (Spreadsheet Auditing by Customs & Excise). In addition to looking for innocent errors, the program also looked for certain types of fraud that the agency had found in earlier audits. For instance, the program looks for a number that was entered as text. Excel treats text cells as having the value zero. Consequently, entering a number in a column of numbers as text reduces the real total. This will create a fraudulent reduction in tax payments.

The All First Fraud Scandal

The most famous case of spreadsheet fraud occurred at Allfirst, which was a U.S. subsidiary of Allied Irish Banks (AIB) in Ireland. The best information about this case comes from the United States Department of Justice (2002).

A currency trader, John Rusnak, began losing money in his trades around 1997. He used a series of spreadsheet subterfuges to hide his losses, which continued to increase. When the fraud was finally discovered, his losses amounted to $691.2 million USD. Although neither Allfirst nor Allied Irish Bank went into receivership, the losses amounted to 60% of AIB's 2001 revenues, and losses produced major drops in stock prices. After the scandal, AIB sold off its Allfirst subsidiary.

AIB immediately commissioned Eugene Ludwig, former U.S. Comptroller of the Currency, to prepare a report on the incident. The story his report tells is an excellent cautionary tale.

Rusnak began by entering two false option trades in the company's trading system-receipt of a large premium and the payment of a large premium. The first option would expire the day of the trade, the other one later. Allfirst had no reports on options that expired the same day and so did not detect what Rusnak was doing. The second option created a false asset on the company's books. This offset the real losing position that Rusnak wished to hide.

Initially, Rusnak used fake broker confirmations to validate his fictitious deals. This was risky because the back office staff reconciled trades with receipts. However, Rusnak convinced back-office personnel that they did not have to confirm the trades because they were offsetting
deals with no transfer of cash.

In 2001, the head of treasury funds at Allfirst noted that Rusnak's trades were using up an unusually large portion of his balance sheet and that this was disproportionate to his earnings. He ordered Rusnak to reduce his exposure on the balance sheet. Rusnak did so, but he accomplished this using highly risk trades that saddled the company with massive potential liabilities.

One control at Allfirst was to compute a value-at-risk (VaR) ratio for each trader. The data for these calculations were supposed to have been computed independent by the back office staff, but Rusnak was able to persuade them to use data on his computer. He manipulated this data to make his VaR ratio look acceptable.

The fraud came apart when a back office supervisor noticed that Rusnak's trades were not being confirmed as required by procedures. The supervisor discovered that a number of trades were clearly bogus. He notified management of the problem. The fraud quickly unraveled.

Rusnak eventually entered into a plea agreement that got him only seven years in jail. This relatively "light" sentence was a result of his agreeing to work with prosecutors in prosecuting people in other companies whose actions prolonged the time it took for Rusnak's scheme to unravel [BBC, 2002].

**Legislation: SOX and Beyond**

**Sarbanes-Oxley (SOX) and Financial Fraud**

We have seen that the Sarbanes-Oxley Act of 2002, also known as SOX, requires U.S. firms and the many foreign firms listed on U.S. stock exchanges to have effective controls for their financial reporting systems and to report on the effectiveness of these controls. For most large firms, the deadline to implement effective controls either has passed or will pass soon.

SOX gives the Securities and Exchange Commission (SEC) overall responsibility for implementing the law. The SEC, in turn, created the Public Company Accounting Oversight Board (PCAOB) to develop specific rules and oversight functions to implement Sarbanes-Oxley.

Although SOX was a response to specific high-profile cases of fraud, fraud in financial reporting has long been a major problem. In 2004, fraud through financial statements represented only 7% of all fraud cases studied in an ACFE survey [ACFE, 2004], but the median fraud loss for financial fraud was a million dollars, compared to a median loss of only $100,000 for frauds in general.

**Other Regulations**

Although SOX has received the most attention, a number of other recent pieces of legislation have required corporations to reconsider their financial systems and other information systems.
SEC Accelerated Filing Deadlines

Beginning in December 2002, the SEC has required firms to reduce the time they may take to produce their quarterly and annual reports. These tighter time limits will make designing controls more difficult because there will be less time to check for errors and violations.

IAS/IFRS

In accounting, U.S. accounting standards are set by the U.S. Federal Accounting Standards Board (FASB), which creates the generally accepted accounting practices (GAAP). In Europe, the International Accounting Standards (IAS) will have to be followed by U.S. firms operating in Europe, including the finance-specific International Financial Reporting Standards (IFRS).

SAS 99

In 2002, shortly after Sarbanes-Oxley was enacted, the Auditing Standards Board produced Statement on Auditing Standards 99 (SAS 99), Consideration of Fraud in a Financial Statement Audit [Auditing Standards Board, 2002]. As its name suggests, this standard requires auditors to search aggressively for fraud. We will look at SAS 99 later in our discussion of fraud control.

Basel II

Banks who do business internationally will also have to comply with the Basel II accord. Basel II requires banks to have capital reserves to cover probable risks. Banks that do not have solid financial reporting systems or risk controls in place must set aside more capital reserves. This reduces the amount of loans they can support, and this in turn limits profits. Basel II gives banks a direct incentive to invest in internal controls to reduce risks.

Privacy Laws

Several laws have affected requirements for privacy and for the disclosure of private information. These include the following:


- Although the E. U. Data Protection Directive is the most important international privacy rule, many other nations with which U.S. firms do business are also developing strong commercial data privacy laws.

- The U.S. Gramm-Leach-Bliley Act (GLBA) of 1999 requires strong data protection in financial institutions.
• The U.S. Health Information Portability and Accountability Act (HIPAA) of 1996 requires strong protection for private data in health care organizations.

• The U.S. Patriot Act of 2001 gives the U.S. government broad powers to see personal data.

The Compliance Age

These laws and many others have made compliance a major issue for information systems in all corporations. Unfortunately, IS education has not kept pace with the growing importance of compliance in IT management.

Control Frameworks

To achieve compliance with SOX and other crucial regulations, companies typically adopt a control framework. Control frameworks help organizations understand all of the things that they need to do and how to do these things.

Types of Controls

All of these frameworks focus on controls. Earlier, we saw in Figure I that the purpose of controls is to help organizations keep their organizational processes on track to achieve the firm's goals. Figure I also shows that controls generally fall into one of three categories.

• Preventive controls attempt to keep deviations from occurring in the first place. In movie theaters, one person sells tickets but another collects them. This is the segregation of duties. Unless the two parties collude, the person accepting the money for tickets cannot collect money, pocket it, and then allow the person in without giving them a ticket.

• Detective controls attempt to detect deviations when they occur, so that action can be taken. Periodic reconciliations between independent processes will make it likely that deviations in one of the processes will be revealed. In the case of movie theaters, management reconciles the number of tickets sold with the number of tickets collected at the end of each day.

• In some cases, there are corrective controls, which fix deviations. The restoration of backup files on a computer compromised by an attack is a corrective control.

This taxonomy is not perfect. For instance, if people realize that detective controls are in place, this should deter them from misbehavior, and this would be a preventative control. However, the general concept of preventative, detective, and corrective controls is useful in practice.

Coso
For Sarbanes-Oxley, the PCAOB explicitly requires corporations to use a well-developed comprehensive control framework. Although the PCAOB does not require corporations to use a specific framework, it has specifically listed only a single framework as acceptable, and most companies are using this framework to implement SOX. This is the COSO framework.

The COSO Framework

Although COSO is universally known by its acronym, the COSO framework actually is a document called Internal Control-Internal Framework [COSO, 1994]. The acronym COSO comes from the organization that created the document, the Committee of Sponsoring Organizations of the Treadway Commission (www.coso.org).

Objectives

Control frameworks, as shown in Figure 1, require objectives. In the COSO framework, there are three objectives.

- Operations. The firm wishes to operate effectively and efficiently. It is necessary for the firm to control its general internal operations to do this.
- Financial Reporting. The firm must create accurate financial reports. This, of course, is the focus of Sarbanes-Oxley.
- Compliance. The firm wishes to be in compliance with external regulation. In this paper, we are only directly concerned with SOX compliance.

Reasonable Assurance

Good controls cannot completely guarantee that goals will be met. However, an effective control environment will give reasonable assurance that goals will be met.

COSO Framework Components

Figure 4 shows the COSO framework. It shows that the framework has five components. These are components rather than phases because there is no time ordering among them. Each must occur simultaneously, and each feeds into others constantly.

Figure 4: The COSO Framework

- Control Environment. The component at the base of the COSO framework is the corporation's control environment. This is the company's overall control culture. It includes the "tone at the top" set by top management, the company's commitment to training employees in the importance of control, the punishment of employees (including senior managers) who violate control rules, attention by the board of directors, and other broad matters. If the broad control environment is weak, other control elements are not
likely to be effective.

- **Risk Assessment.** More specifically, a company needs to assess the risks that it faces. Without systematic risk analysis, it is impossible to understand what level of controls to apply to individual assets. Risk assessment must be an ongoing preoccupation for the firm because the risk environment constantly changes.

- **Control Activities.** An organization will spend most of its control effort on the control activities that actually implement and maintain controls. This includes such things as approvals and authorization, IT security, the separation of duties, and other matters we will look at later. Controls usually have two elements. One is a general policy, which tells what must be done. The other is procedures, which tell how to do it.

- **Monitoring.** Having controls in place means nothing if organizations do not monitor and enforce them. Monitoring includes both human vigilance and audit trails in information technology. It is essential to have an independent monitoring function that is free to report on problems even if these problems deal with senior management.

- **Information and Communication.** For the control environment, risk assessment, control activities, and monitoring to work well, the company needs to ensure that it has the required information and communication across all levels of the corporations.

**Types of Control Activities**

Internal Control—Internal Framework [COSO, 1994] does not list a comprehensive set of control activities, probably because it is impossible to create a complete list of possible controls. However, the document does provide several lists of types of control activities. For instance, on Page 49, the framework notes the existence of manual controls, computer controls, and management controls. On the following page, it provides the following list to consider:

- **Top Level Review**—comparing budgets with actual performance, tightly monitoring major initiatives, and so forth.

- **Direct Functional or Activity Management**—managers who run individual operations must examine the appropriate reports for their level, for instance, loan performance in a bank's lending operations.

- **Information Processing**—including the enforcement of manual procedures, such as checking if a customer's accounts payable are not above a certain amount before accepting an order. Note that information processing must focus on business processes, not merely on IT processes.

- **Physical Controls**—inventories, locked cash drawers, write-only archival media, and so forth.
Performance Indicators-Relating different sets of data to each other to check for inconsistencies, noting deviations from normal performance (in either direction), unusual trends, and so forth.

Segregation of Duties - requiring most sensitive processes to be completed by two or more people, so that no single person can engage in improper activities without this becoming apparent. For instance, one person may purchase and order, but another person will record it. Alternatively, no single person can both authorize and make a purchase.

Controls for Information Systems

On Pages 52-55, the document lists some controls over information systems. At a most basic level, the framework discusses the differences between application controls and general controls. Application controls, as the name suggests, involve individual applications (accounting applications, spreadsheets, and so forth), including manual operations in using them. This includes interfaces to other systems for input data, checks on input, internal checks during processing to flag errors and misbehavior, and so forth.

General controls cover everything beneath the applications-computers, operating systems, the network, and so forth, together with manual operations in using them. This includes making purchases, application systems development, maintenance, access controls, evaluating packets software, and so forth. The controls needed in individual applications will depend on the quality of general controls.

Controls over "Evolving Issues"

The report spends approximately half of Page 55 on "evolving issues." Two brief paragraphs are devoted to end-user computing (EUC). The first paragraph simply says that EUC exists. The second give the following meager guidance:

To provide needed control for EUC systems, entity-wide policies for system development, maintenance, and operation should be implemented and enforced. Local processing environments should be governed by a level of control activities similar to the more traditional mainframe environment. (COSO, 1994, p. 55)

Internal Control-Integrated framework does not give any specific guidance on spreadsheets. In fact, it does not mention them. In general, this is an old (1994) document that was written before spreadsheets become important, or, probably more correctly, before IT control professionals realized that spreadsheets were important.

CobiT

COSO is a general control planning and assessment tool for corporations. For IT controls, there is a more specific framework, CobiT (Control Objectives for Information and Related Technologies) (IT Governance Institute, 2000). The IT Governance Institute has not only created the control objectives framework. The institute also has developed detailed guidance for
implementing the CobiT framework.

The CobiT Framework

Figure 5 illustrates the CobiT framework. This framework has four major domains:

**Figure 5: COSO/CobiT Framework**

- **Planning and Organization.** The four domains follow the IT life cycle. The planning and organization domain has 11 high-level control objectives that cover everything from strategic IP planning and the creation of a corporate information architecture to the planning of specific projects.

- **Acquisition and Implementation.** After planning, companies need to acquire and implement information systems. This domain has six high-level control objectives.

- **Delivery and Support.** Most of an IT project's life takes place after it implementation. Consequently, the CobiT framework has 13 high-level control objectives for delivery and support. This is more than any other domain.

- **Monitoring.** Finally, firms must monitor their processes, assessing the adequacy of internal controls, obtaining independent assurance, and providing for independent audit.

Although the four domains define the scope of CobiT, they are only the beginning of CobiT. Beneath the four domains are 34 high-level control objectives, which Figure 5 also shows. Beneath these, in turn, are more than 300 detailed control objectives. CobiT also includes many documents that help organizations understand how to implement the framework.

**Dominance in the United States**

The IT Governance Institute is the child of the Information Systems Audit and Control Association (ISACA). ISACA, in turn, is the primary professional association for IT audit professionals in the United States. The association's certified information systems auditor (CISA) certification is the dominant certification for U.S. IS auditors, so it is not surprising that CobiT has become the dominant framework for auditing IT controls in the United States.

**COSO and CobiT**

Obviously, both COSO and CobiT pertain to information technology used in financial reporting. Figure 5 shows how CobiT relates to COSO at a broad level. This figure illustrates that it is relatively simple to combine COSO with CobiT at a conceptual level, although the details are anything but simple.

**Other Frameworks**

Although COSO and CobiT have dominated Sarbanes-Oxley planning in the United
States, other planning frameworks are also important.

ISO/IEC 17799

Figure 6 shows the relationship between COSO, CobiT and three other frameworks-ISO/IEC 17799, Common Criteria, and ITIL. The figure emphasizes that the three frameworks overlap but focus on somewhat different things.

Figure 6: COSO, CobiT, 17799, Common Criteria, and ITIL

For example, CobiT, as its name implies, focuses specifically on controlling the entire IT process, while COSO focuses more broadly on financial reporting controls. In contrast, ISO/IEC 17799, Code of Practice for Information Security Practice (ISO/IEC, 2000), as its name suggests, focuses more narrowly, on IT security. Security is part of IT controls, of course, so 17799 can help in creating IT controls.

ISO/IEC 17799 grew out of an earlier standards effort by the British Standards Institute. In 1995, the Institute produced BS 7999. This standard has two parts. ISO and the IEC adopted Part I as 17799. This first part is a broad code of practice.

For organizations wishing certification of their standards effort, Part 11 of 7999 (Information Security Management System) has auditable controls. Consequently, many companies have chosen to be compliant with 17799 by being certified in Part 11 of 7999.

In other frameworks, including COSO and CobiT, companies certify themselves, sometimes with the concurrence of an external auditor. They lack 17799's third-party certification process, which external parties may value highly.

Even 17799 is quite complex. Broadly, it has ten major categories' of standards. It subdivides these into 127 specific controls. At the time of this writing, 17799 is in flux. ISO is preparing to release a new version in mid-2005. Although there has been much speculation about the revisions in the standard, it seems best to hold comments until the final 2005 version. In addition, it seems likely that ISO and IEC will adopt BS 7999 Part 11 as an ISO standard. Again, however, we must wait for ISO and IEC to act.

Common Criteria

Figure 6 shows that the Common Criteria (ISO/IEC 15408, Information technology -- Security techniques -- Evaluation criteria for IT security) standard is even more specific, Common Criteria specifically focuses on the evaluation of security products, such as firewalls. It provides a way for purchasers to know specifically what security features a security product claims to offer and how rigorously the product was developed. However, the Common Criteria approach has somewhat limited use because it is difficult to apply and does not provide a high level of assurance that a product actually is secure.

ITIL
Another framework for IT is the Information Technology Infrastructure Library (ITIL). ITIL is a broad set of best practices guidelines for providing IT services. It is widely used in Europe and is becoming popular elsewhere. ITIL is highly process-oriented, specifying systematic approaches to implementing security and other IT services. ITIL best practices may be helpful in implementing other guidelines. However, ITIL does not have the detailed guidance necessary for developing and implementing IT financial reporting controls.


Spreadsheet-Specific Guidance

If one does an Internet search with the terms "spreadsheet" and "Sarbanes," there will be many hits. Nearly all of these, however, are about spreadsheets used to document SOX compliance, not how to control spreadsheets used in financial reporting.

The one major exception to this silence on spreadsheet control for Sarbanes-Oxley is a six-page report by PriceWaterhouseCoopers [2004]. This report lists a large number of controls. The first step is to inventory all of a firm's spreadsheets that are "in scope," that is, are used in financial reporting. Next steps are to evaluate the riskiness of these spreadsheets, determine necessary controls, evaluate the existing (as-is) controls on these spreadsheets, and develop action plans for remediating control deficiencies. Once concern is the report's method for assessing riskiness. It lists nine factors to consider when evaluating the "risk and significance" of a spreadsheet:

- Complexity of the spreadsheet and calculations
- Purpose and use of the spreadsheet
- Number of spreadsheet users
- Type of potential input, logic, and interface errors
- Size of the spreadsheet
- Degree of understanding and documentation of the spreadsheet requirements by the developer
- Uses of the spreadsheet's output
- Frequency and extent of changes and modifications to the spreadsheet
- Development, developer (and training) and testing of the spreadsheet before it is utilized

Although this is a good list, it appears to be based on an incorrect view of spreadsheet errors. Research indicates that the largest indicator of the number of errors in a spreadsheet is
simply the length of the spreadsheet. There will be errors in about 2% of all non-null cells if there is no deep testing. This also is the case in programming. However, from programming, we know that more complex programs have more errors than simpler programs, but only by a factor of about four [Zage & Zage, 1993]. Consequently, long simple spreadsheets will have many more errors than short complex programs. Certainly, reporting complexity first and size halfway down the list is a concern.

Even more of a concern is placing testing last in the list. As discussed in the following sections, a lack of testing is a devastating control deficiency, not simply one part of a last concern on a list.

The report also lists a number of controls that should be considered to mitigate risks inherent in the spreadsheet environment.

- Change control: the authorization of change requests, testing the spreadsheet, and formal sign-off by another individual.
- Version control: Ensuring that only the current and approved version of each spreadsheet is used. Naming conventions that include version numbers and dates and the use of structured directories can help in this.
- Access control: assign appropriate access rights to people who need to use the spreadsheet. Use a password to control access.
- Input: Whether manual or automatic data entry is used, there should be reconciliations.
- Security and data integrity: storing spreadsheets in protected directories, and locking formula cells to prevent logic changes.
- Documentation: Ensure that the business objective and functions of the spreadsheets are understandable.
- Development life cycle: Use a standard systems development life cycle. The report specifically says that testing is critical (but does not discuss how to do testing).
- Backup and Archiving: spreadsheets should be backed up because of their sensitivity. They should be archived in read-only format for later review.
- Logic inspection by an independent person other than the developer: The report does not discuss how logic inspection is different from testing. Probably, testing is taken as what we call execution testing later in this paper. The report appears to sanction logic checking by a single individual. As discussed later, this appears to be unwise.
- Segregation of duties/roles, and procedures: There should be authorities, roles, and procedures for ownership, sign-off, and other matters. This item is so brief in the report that it provides little guidance.
Analytics. In accounting, "analytics" refers to calculations built into spreadsheets (ratios, cross-checks, statistical patterns, etc.) that help detect errors and fraud.

The report says that firms should enforce these controls. For instance, changes should be made independently in two copies of each spreadsheet, and the two copies should be compared. In another example, a sample of cells can be tested to ensure that they are password-protected if they should be. In addition, names should include modification dates, and names should be compared with modification dates as recorded by the operating system.

Finally, the report gives a number of suggestions for remediation. Specific responsibilities should be assigned to specific people, remediation efforts should be prioritized, and remediation deadlines should be established.

Although this report offers valuable advice, its error beliefs are inadequately, and its lack of detail leave many questions unanswered. One important concern is testing. We will discuss testing in some detail in this paper. Another issue is how to control manual processes in spreadsheet usage.

The Testing Control

Although spreadsheets need many controls, we will focus on testing because of testing's extreme importance in software quality assurance.

Although fault rates in programming are very similar to error rates in spreadsheet models, programmers and spreadsheet developers react very differently to these defect rates. Programmers, appalled by their fault rates, spend a great deal of time on programming. In a sample of 84 projects in 27 organizations, Jones [1998] found that the amount of time spent in testing to reduce errors ranged from 27% to 34% depending on program difficulty. In every case, subjects reported that insufficient time was planned for testing (p. 139). In another study, Kimberland [2004] found that Microsoft software development teams spent 40% to 60% of their total working time in testing.

The situation in spreadsheet development is very different. As noted earlier, studies since the earliest days of spreadsheet development have found that few organizations have policies requiring extensive testing and that few spreadsheet developers test their spreadsheets intensively in ways similar to those that programmers use.

Testing in Software Development

Both field audits [see Figure 3] and experiments [Panko 2005b] have shown that spreadsheet models and software programs are very similar in error rates and error types. Consequently, to get insight into spreadsheet testing, it would be good to understand software testing.
Execution Testing

There are two general ways to test software for errors. One is to use execution testing, in which the tester assembles a large number of test values with known results and runs these test numbers through the program as input values. We will discuss execution testing again under spreadsheet testing.

Code Inspection

Another method is code inspection, which Fagin [1976] developed. In code inspection, a group of testers independently examines a program line by line looking for errors. They then meet to compare their results and attempt to find new errors. Although variations on this basic pattern exist, all code inspection methods involve groupwork and line-by-line code examination.

One tenet of code inspection methodologies is that testers should report their results. The programming inspection data in Panko [2005a] are extensive because so many code inspection studies have reported their results. This reporting is important because it forces software developers to face error rates directly.

Code Inspection Parameters

Fagin [1976], based on his personal experiences implementing code inspection, emphasized several principles for code inspection. Subsequent experience has reinforced his insights.

- Module Size. Perhaps the most important thing is to test small modules rather than entire programs. A typical module size is 100 to 200 lines of code. Although there disagreement about the optimal module size, there is no doubt that keeping modules small is important. For instance, Bernard and Price [1994] found that code inspectors found 72% more errors when modules were shorter. If the module is too long, the tester will not be able to maintain his or her focus during testing.

- Inspection Rates. Another key principle is that inspections should not be rushed. Inspection rates often are measured as lines of code inspected per hour. Basili and Pericone [1993] found that when the inspection rate rose from 50 lines per hour to 200 lines per hour, the detected fault rate fell from 1.6% to only 0.6%. This was not because programs had fewer faults but because faster inspection detected fewer errors. Russell [1991] found an even sharper drop in detection when the inspection rate rose from 150 lines per hour to 750 lines per hour. Ebenau and Strauss [1994] found that "hasty" code inspections found errors in 1.3% of the lines of code they examined, while non-hasty code inspection found errors in 2.0% of all lines of code. Weller [1993] also found sharp drops in detection rates as inspection rates increased. Levy and Begin [1984], in research on proofreading, similarly found that faster proofreading speed substantially reduced the percentage of spelling errors detected.
• Formula Inspection Speed. One thing that has not been studied in the code inspection literature but that probably is important is the need to slow down even further for complex equations. In proofreading, Healey [1980] looked at error detection for nonword errors (errors that do create a dictionary word) in prose for varying word lengths. Detection rates were over 90% for words of 2, 3 or 4 letters. For longer words, however, detection rates were only about 75%. Panko [1999], in a study of spreadsheet code inspection, found that error detection rates for long formulas were much lower than for shorter formulas. Just as drivers should slow down for hazardous road conditions but rarely do [e.g., Howarth, 1988; Svenson, 1977], code inspectors need to force themselves to slow down for long formulas.

• Team Size. Another tenet of code inspection is that code inspection must be done by groups rather than by single inspectors. Weller [1993], who analyzed more than 6,000 code inspection studies, found that teams of four found 29% to 108% more errors than teams of three depending on the inspection rate. This limited ability of individuals to find errors also been seen in laboratory code inspection experiments. In these experiments, subjects found only 22% [Johnson & Tjahjono, 1997], a third [Myers, 1978] and a half [Basili & Selby, 1986] of the programming faults seeded in a test program. When Johnson and Tjahjono [1997] had subjects work in groups of three, detection rate doubled. In two spreadsheet code inspection studies, Galletta et al. [1992, 1997] found that individual detection rates for a spreadsheet seeded with errors were only 51% and 66% in the two studies. In another study using a similar spreadsheet [Panko, 1999], individuals only found 63% of all errors, while groups of three found 83%. Although this seems like a modest increase, this increase came precisely in the errors that individuals were least likely to detect. Steiner [1971] developed a theory of why groupwork is effective. He noted that groupwork is more important for difficult problems than for easy problems.

• Team Composition. It is sometimes said that people who develop code should not be on the inspection team because people are not good at finding their own errors. However, proofreading research suggests differently. Daneman and Stainton [1993] found that when proofreading took place twenty minutes after writing, individuals found only 59% of their own word errors but found 76% of word errors made by others. However, when proofreading took place two weeks later, people had higher success reading their own work than did others (83% rather than 68%). If inspection takes place some time after development, individuals should be able to be effective at code inspecting their own work, although this hypothesis has not been tested in programming or spreadsheet development.

• Reasonable Expectations for Detection Yields. The reason for the use of team code inspection is that humans are only moderately good at finding errors. This has been seen in many areas of human cognition, such as proofreading [Panko 2005a]. Team inspection can correct more errors, but even team inspection is far from perfect. Jones [1988], based on extensive experience, estimated that each formal code inspection only detects about 60% of all errors (p. 199). McCormick [1983] found that code inspections he studied found 86% of all errors, based on the number of errors found later. Boehm and Basili [2001], based on their experiences with many studies, reported that peer reviews only caught a median of 60% of all errors and that this yield ranged from 31% to 89%. At a CeBASE [2001] workshop, participants shared data from several studies, collectively indicating that 60% or
more defects usually are found in inspections. Due to limited error detection rates, programming projects have to have several rounds of inspection. Studies of late inspections after multiple rounds of inspections still find errors in 0.1% to 0.3% of all lines of code [Putnam & Myers, 1992]. Given that most studies of initial code inspection find errors in 2% of all lines of code, this data suggests that even multiple rounds of code inspection only find about 90% of all errors.

- Improvement Takes Time. It would be nice if companies could learn to do code inspection quickly. However, experience has shown that improvement is likely to take two or more years [Haley 1996, Spencer 1993]. In general, it takes considerable time for people to become competent in complex skills. For instance, it takes about two years of driving before people cease being menaces on the highway, [Bereiter & Scardamalia, 1993]. In addition, once people are skilled, it may be a considerable period of time before the organization becomes competent using new methods.

Testing Spreadsheets

From the information just presented, it is clear that spreadsheet errors are a serious threat in all large spreadsheets. In software development, where fault rates are similar to error rates in spreadsheet development, most companies mandate a very disciplined development process. This process involves several well-controlled stages, including requirements definition, design, coding, and testing. All of these stages are important, but the biggest reductions in error rates in programming come from testing. This includes both design reviews and coding tests, but we will focus on coding tests. As noted earlier, approximately 30% of all software project effort is devoted to testing.

Whiskey Cures

Ineffective home remedies sometimes are called "whiskey cures," after the old aphorism, "Of all the things that do not cure the common cold, whiskey is the most popular." With a whiskey cure, you take a step to reduce harm, but the step is largely or entirely ineffective. One commonly seen activity in spreadsheet development is looking over a spreadsheet's results for reasonableness [Hendry & Green, 1994; Nardi & Miller, 1991]. If a few errors are found, the seeker feels that he or she is very effective at finding errors. If no errors are found, then the seeker feels that the spreadsheet is free of error. Unfortunately, given the difficulty of finding errors noted above even when full cell-by-cell code inspection is used, merely looking over a spreadsheet's numbers for reasonableness is a whiskey cure. In addition, studies [Klein, 1997; Rickets, 1990] have shown that people are not very good at finding errors when they assess numbers for correctness. Although looking over results for reasonableness is simple and inexpensive and should be done, it must not be considered an acceptable stopping point.

Another whiskey cure is depending upon developers to test their own spreadsheets during development. However, testing during development is neither new nor very effective. When Allwood [1984] watched students working on algebra problems, he noted that they frequently stopped to go back and check their work. In some cases, they did this when they thought they had made an error. In other cases, they performed what Allwood called a standard check, going
back over their work on general principles. Hayes and Flower [1980] saw that when people write they constantly shift among planning, writing, and reviewing activities. Reviewing is like the standard check in Allwood's study. Kellog [1994] found that people spend 20% to 25% of their time of their writing time reviewing. In an earlier study cited in his 1994 paper, Kellog found that subjects spent 19% of their time reviewing and also found that reviewing increased as writing time increased.

In spreadsheeting, Panko and Halverson [1997] also watched many of their individual subjects and groups during spreadsheet development. They also noted frequent error checking during development but only rare post-development testing. However, many errors remained undetected despite this type of review during development.

**Execution Testing**

A common form of software testing is execution testing, in which the tester inputs values that have known output and observes whether the program gives the correct results. This approach seems obvious, but it often is very difficult to do well. In many cases, the spreadsheet is doing calculations that have never been done before. Consequently, finding the correct output values for test cases may not be feasible or may be prohibitively difficult.

In addition, determining proper test cases is a subtle skill. Most people tend to pick test cases that will prove a program to be correct. This is in line with the human confirmation bias [Snyder & Campbell 1980]. Proper execution testing, however, requires the tester to select cases that are likely to break the program. This requires a great deal of training and experience. Unfortunately, spreadsheet developers rarely have the proper training and experience.

Overall, execution testing in spreadsheet development is likely to be most effective for testing changes to spreadsheets. After a spreadsheet is developed and tested, test cases should be developed, and the actual values should be taken as correct values. If the spreadsheet is modified afterwards, the test cases should be rerun. Variables that should not change due to the modifications should be the same as the original values. If they are not, this will indicate an error.

**Code Inspection**

Although spreadsheet developers are not likely to be good at developing test cases for execution testing, they are likely to be quite good at code inspection. They usually are subject experts in the content domain of the spreadsheet. This means that they are likely to understand both the formulas in the spreadsheet and the flow of logic among the formula cells.

Given the threat analysis presented earlier and the discussion of software code inspection presented earlier, spreadsheet code inspection should have the following characteristics. This is not a complete list, but it emphasizes the points that are likely to be the most difficult to implement.

- **Inspect All Cells.** There is no excuse to say that code inspecting all cells is impractical. Not doing it guarantees an unacceptable error rate. In addition, if programmers can do it,
spreadsheet developers can do it. Unless code inspection inspects all cells, it is a whiskey
cure.

- Use Group Inspection. A single person will be unable to find many errors in a
  spreadsheet. The only question is how large the group should be. As noted earlier, Panko
  [1999] found that the detection rate rose from 63% to 83% when individuals gave way to
  groups of three. If this finding holds, then larger groups will be desirable.

- Inspect Modules. Trying to inspect an entire spreadsheet is a guaranteed route to failure.
  Inspections should be conducted on each module after it is developed. The optimum
  module size is unknown, but in Panko's [1999] experiment, a spreadsheet with only
  about 20 unique formulas took students approximately 45 minutes to inspect. A
  maximum inspection time for a module is about two hours if attention is to be preserved,
  so optimum module sizes are likely to be fairly small. Most spreadsheets will have many
  modules and so will require many inspections. The optimum size in terms of cost and
  detection rates needs to be determined empirically.

- Slow down for complex formulas. As noted earlier, Panko's [1999] study showed that
  inspection success falls for cells that contain longer formulas. It seems likely that when
  inspectors reach longer formulas, they should slow down their testing rate dramatically.

- Multiple Rounds of Inspection. If program inspection is a good indicator, each round of
  inspection is only likely to detect 60% to 80% of all errors, and achieving even these
  levels is likely to require considerable experience.

**Actual Testing**

How do real-world companies do testing? As noted earlier, when the first author asked
financial professionals and officers whether their firms did testing for material spreadsheet, most
did not, but a small subset said that they did. However, the next question asked respondents how
they actually tested their spreadsheets of material importance in financial reporting. As Figure 7
shows, 73% of the respondents said that when their firms test spreadsheets of material
importance, they test only some cells. In other words, they consider "looking the spreadsheet
over" to be testing. Only 12% said that their firm tested all cells in the spreadsheet, and only 2%
said that their firm used both multiple testers and cell-by-cell testing.

**Figure 7: Testing Methods for Material Financial Spreadsheets**

**Spreadsheet Auditing Programs**

One possible approach to testing is to use spreadsheet auditing programs that are designed
to find errors in spreadsheet programs automatically. Obviously, using a spreadsheet auditing
program is faster and less expensive than manual code inspection.

Unfortunately, to borrow a term used in U.S. prescription drug regulations, spreadsheet
auditing programs have not been proven to be "safe and effective." No research known to the
authors has established what percentage of errors auditing programs will find. Unless this percentage is very high, it will not be safe to use spreadsheet auditing programs in lieu of full manual code inspection.

To give an analogy, spelling checkers in word processing programs will catch all spelling errors in which the mistyped word is not a dictionary word. However, if the word is in the dictionary, the spelling checker will ignore it completely. Proofreading research has shown that the former type of error is, also easiest type of spelling error for humans to detect. Grammar checkers, in turn, are frustrating for missing grammar errors and for calling correct grammar incorrect.

We suspect that auditing programs will be even less precise than grammar checkers. They are unlikely to be able to find omission errors, such as if the developer leaves out paying the monthly rent. Nor are they likely to be able to catch most logical formula errors based on the use of the wrong algorithm or the incorrect expression of an algorithm. Errors in typing numbers are also likely to escape these programs. It seems like they would be best in pointing errors, but these are quite uncommon [Panko & Halverson, 1997, 2000]. In programming, Endress [1975] found that 48% of all errors are problem misunderstandings. In spreadsheet development, auditing programs are not likely to be effective with such errors.

In general, because there are multiple types of errors, almost any error reduction or detection method is likely to be limited. Beizer's [1990] Pesticide Paradox reflects this problem of multiplicity in dealing with errors in the context of programming: "Every method you use to prevent or find bugs leaves a residue of subtler bugs against which those methods are ineffective."

In marketing materials, spreadsheet auditing programs focus on the ability to detect patterns in how blocks of cells are laid out and on certain types of link geometries. Panko [2000] has listed the most common errors for two spreadsheet development tasks. Few of these errors seem amenable to detection based on geometry.

The most common advice to writers is to use automated spelling and grammar checking functions for "pre-proofing" of a document before full manual proofreading is done. The same may be the case with automated spreadsheet auditing programs.

Fortunately, Excel has a number of pre-auditing tools that can be used by everyone. Excel tags many questionable formulas with green comment markers. By going to Tools, Error Checking, the user can step through these potential errors in order. One choice usually is Evaluate Formula, which steps the auditor through a formula's calculation term by term.

In addition, for manual code inspection, a user can see the precedents of a formula (cells to which a formula refers) by double-clicking on a formula cell or clicking on the cell and hitting F2. Cell references and the cells or ranges to which they refer are shown in the same color.

By bringing up the formula auditing toolbar (View, Tool Bars, Formula Auditing) the user will have several options, including showing subsequent formulas that refer to a particular
formula (decedents). A cell with no decedents has to be a bottom-line value. Any cell in a spreadsheet that is not a bottom-line value yet has no decedents is suspicious.

Other Error Controls

Although we have focused on testing as a sine qua non for reducing errors, other controls are needed to achieve low error rates.

Design and Design Reviews

In programming, the standard systems development life cycle calls for strong initial planning, in which the systems analyst determines needs, in which a programmer creates a design for the program, and in which the organization conducts a design review. Errors found during design reviews are much less expensive to fix than errors in later phases, especially coding. Studies [McCormick, 1983; Jones, 1986] have found that more errors occur during the requirements and design stages than during the coding phase, so design reviews should be able to reduce error reduction costs considerably.

Unfortunately, not only is design review rare in spreadsheet development but design itself tends to be limited. In experiments, observations of people developing reasonably challenging spreadsheets has shown that almost all subjects begin entering labels, numbers, and formulas within a minute or two at most [Brown & Gould 1987, Panko & Halverson 1997, Takaki 2005].

Given the cost effectiveness of design reviews and the danger of creating difficult-to-understand spreadsheets if there is insufficient planning, companies need to develop mandatory design and design review processes for all nontrivial spreadsheets.

At the same time, when people develop spreadsheets, they may not understand the situation fully. The process of building the spreadsheet model itself may be the best way to understand the situation [Panko 1988] because the developer can work until hitting a snag that must be addressed or because the developer may begin to develop a spreadsheet and realize that the initial design will not work. In addition, as the user begins to do calculations, he or she might find surprising results. These results can lead to new directions in design. Similarly, in writing, Kellog [1994] noted that inspiration generates production, which generates more inspiration.

Of course, spreadsheets that grow "organically" may have many design and construction flaws. One possibility is to use such spreadsheets as prototypes and then to build production systems from scratch, using the prototype's general design. However, this takes a great deal of discipline. In writing, Flower and Hayes [1980] have suggested that writers should have "plans that are detailed enough to test, but cheap enough to throw away." The same should be true for spreadsheet developers.

No matter how the overall spreadsheet is designed, it seems important to design individual formulas before constructing them. The first author's observations of laboratory subjects developing spreadsheets indicated that they often entered a formula.
they were not sure of and then looked at the results to see if they looked right. This "ready, fire, aim" approach is fast, but it is not safe.

Formula Protection

As noted earlier, users sometimes "hardwire" spreadsheets by typing numbers into formula cells. Dent [1994] reported that hardwiring was very common in a mining firm. In another case, Rabbit Photo in Australia uses a spreadsheet program for budget and expense consolidation. The company found that users were constantly overwriting formula cells to enter "correct" values. When the company turned on cell protection to prevent the overwriting of formulas, users turned off protection and made their changes anyway. The company eventually moved to a centralized software package with a spreadsheet-like interface. Users could not overwrite formulas in this program. Today, companies can use passwords to prevent users from turning off cell protection, but this will require the selection of good passwords. Given the wide extent of the hardwiring problem, companies need to address cell protection actively.

Readability

In a badly constructed spreadsheet, readability will be so poor that testing will be extremely difficult. It is important to implement controls to ensure readability.

- Modules. Testing requires reasonable-sized modules. Consequently, it is important to develop spreadsheet models as a group of modules. This can also improve readability.

- Flow. In most countries, we read from left to right and then from top to bottom. It is important for readability to have this type of clear basic flow in the spreadsheet's logic. Boehm and Basili [2001] found that good architecture in programming can reduce error correction time dramatically. In spreadsheets, Saariluoma [1989] found that poor logic flow could dramatically increase the time it took to understand spreadsheets.

- Short Arcs of Precedence. Raffensperger (2003) has given a number of suggestions to improve spreadsheet readability. One is to use short arcs of precedence between each formula and the cells that precede it. This requires formatting to be simple so that there is not too much vertical white space. Keeping arcs of precedence also is consistent with spreadsheet error research. Lerch [1988] found that people often make errors when entering formulas that refer to distant cells. Janvrin and Morrison [1996, 2000], in turn, found that the error rate was extremely high when links had to be made to data or formulas in other worksheets in a spreadsheet workbook.

- Panko [1988] suggested two basic principles for entering formulas. One was to avoid jamming, in which a cell formula contains multiple numbers. For instance, if a company sells 500 units of a product at $400, the developer might enter a cell with the content "=500*400." This is jamming. It is better to have one cell labeled units and containing the number 500 and a second cell labeled price with the value $400. The revenue cell, then, would refer to the other two cells. In short, there should be a single number in a cell or no
numbers at all. Teo and Tan [1997] confirmed that jamming leads to increased errors during subsequent what-if analysis. With jamming, it is difficult to know which cells to change because individual numbers do not have text labels to identify them. Avoiding jamming also improves readability because readers can understand the labeled numbers more easily. For this same reason, avoiding jamming also makes testing easier. Jamming does have one advantage: it shortens arcs of precedence. However, this is a small benefit compared to jamming's loss of readability.

- Panko [1988] also suggested that numbers should not be entered more than once. For instance, if units sold is used in both revenue and cost calculations, the number of units sold should be entered into a cell only once and then referred to in revenue and cost formula cells. Lukasic [1998] examined two large spreadsheets and found that the same number was often entered multiple times and that the number was not always changed consistently in what-if analyses.

- Repeat numbers across multiple columns through referencing. Teo and Tan [1997] did find one situation in which repetition is valuable. In a time-period-based spreadsheet, when students entered numbers like tax rates in all columns of a spreadsheet, they made fewer errors during what-if analysis than if only a single cell had the tax rate. This makes sense because Lerch [1988] found that when people entered references to cells in different rows and columns they made many more mistakes than they did when referencing cells in the same row or the same column as the formula cell. Even so, the actual number should only be typed once and then referenced in other cells in the row that contains it.

**Appropriate Documentation**

Almost all spreadsheets have known limitations. There appears to be a need to have a documentation worksheet that gives readers the context of the spreadsheet's development. For instance, the documentation should warn that certain information had to be estimated because time pressures prevented the collection of data. In addition, if input values only have limited ranges over which the spreadsheet has been tested, this should be noted as well.

**Controversies**

- The controls just noted all have a foundation in research findings. Several other prescriptions are often given. Unfortunately, they either have no research basis or appear to run counter to research findings.

- Use an Input/Processing/Output Design? A very common prescription is for each module to have an input section containing data, a processing section showing intermediate formulas, and an output section containing the results. There is no research supporting this approach at all. In addition, this approach creates extremely long arcs of precedence, which are very likely to lead to pointing errors when entering formulas. This approach is also likely to harm readability. To understand calculations, the reader
would have to switch constantly between the three sections. Note that most tax preparation programs for individuals show calculations with the data the calculations use. This provides high readability.

- **Use Long Formulas?** Although Raffensperger [2003] has made many valuable suggestions for improving readability, one of his suggestions appears to run counter to research. This is his suggestion to use complex formulas rather than a series of several simpler formulas. As noted earlier, proofreading research shows that people's error detection rate for spelling errors falls rapidly with word size. Panko [1999] found that this was also true for spreadsheet formulas.

- **Clarity Rather than "Efficiency."** It is important to make logic flows as clear as possible, even if there is a shorter way to express logic. Given the speed of today's computer, there is no excuse for "tricky coding." Readability of logic flows is paramount in avoiding errors during development as well as in enhancing testing.

- **Use Range Names?** Another common suggestion is that developers should use range names in their formulas. Although this appears to be attractive, it raises a serious concern. In selecting ranges for range names, pointing errors can assign the wrong range to a range name. Although the range will be wrong, but it will appear to be correct in formulas that merely reference the range names. Although the research findings are not clear on this issue, using range names should be considered potentially dangerous until research on using range names is done.

- **Appropriate Code Documentation.** Almost everyone suggests that documentation is important, but it is rarely done in practice because the task is so onerous. One potential option is use code documentation only where the flow of logic is likely to be non-obvious. This could be done with comments inserted into formula cells. This approach might reduce code documentation sufficiently to increase its use. Research has not been done in this area.

- **Risk Analysis.** A number of prescriptions now call for risk analysis, in which the likelihood of errors and their likely impacts are assessed. Riskier spreadsheets should have more controls. Although the need for risk analysis is obvious, most methodologies have poor approaches to assessing the likelihood of errors.

- **Spreadsheet error research uniformly indicates** that the main factor leading to the likelihood of error is the length of the spreadsheet in terms of calculation chains. Methods tend to focus too much on formula complexity and other factors that are likely to be secondary. It seems mandatory to assume a 2% cell error rate in any risk analysis and to assume, consistent with software research, that 10% to 20% of all errors will be serious.

**Controlling for Fraud**
Traditionally, spreadsheet error researchers have focused on "honest error," that is, the assumption that incorrect spreadsheets result from honest mistakes. However, people who commit fraud may use spreadsheets to execute their frauds. Controlling for fraud in spreadsheet development requires new approaches to controlling spreadsheet usage in the firm. Given the importance of Sarbanes-Oxley, controlling for fraud involving spreadsheets has become a serious concern.

**Fraud**

We discussed fraud informally earlier. Legally, a fraud exists when one person knowingly lies or conceals information whose nondisclosure would make other statements misleading, in order to get the victim to act in a way contrary to the victim's interests. Note that two elements are needed for there to be fraud: deception and harm. As noted earlier, when people develop spreadsheets, there often is bias that does not rise to the level of fraud. However, when the impact becomes large, then fraud occurs.

**The Fraud Triangle**

As noted earlier, the Auditing Standards Board's SAS 99 requires auditors to be extremely vigilant about fraud. Compared to SAS 82, which the new standard replaced, SAS 99 provides much more guidance on the detection of fraud, and SAS 99 requires many more mandatory actions to seek out fraud. SAS 99 requires significant changes in mandatory auditing procedures and documentation in a financial statement audits.

The standard requires a mindset that constantly asks questions with a critical mind in order to identify risks and actual fraud. The standard requires the auditor to look for political promises that will be difficult to meet legitimately, budget pressures, general management style, the effectiveness of the firm's audit committee, the vigilance of the board of directors, and many other things.

One contribution of SAS 99 is the fraud triangle, which is shown as Figure 8. The fraud triangle emphasizes that auditors need to look for three indications of fraud risk.

**Figure 8: The Fraud Triangle**

- **Incentive.** Auditors must constantly ask why employees would be attracted to fraud in particular circumstance. An obvious motive is the misappropriation of assets, but other motivations are very common. For instance, in many cases, the goal of the fraud is to cover poor performance (as was the case at Allfirst) or to make a firm look like it is doing better than it actually is to attract more capital.

- **Opportunity.** Even if incentives are strong, if there is little opportunity to succeed, then the existence of strong controls will deter most fraud. Conversely, if controls are weak, this will offer a strong attraction to employees with potentially criminal motives.
• Rationalization. Law enforcement officials have long known that it is important to understand how criminals justify their behavior to themselves. For instance, when the fraud is done to cover up poor performance, the perpetrator is likely to believe that they will "make things good" in the future. In other cases, there is a rationalization that "everybody does it," or "the company owes it to me." A strong control environment with intensive ethics training can help reduce rationalizations. It is very important for auditors to understand these motivations and opportunities so that they can accurately assess the potential for fraud in given situations.

Fraud Controls

Fraud controls are ways to reduce the frequency, severity, or difficulty of detecting frauds. In general, there are four general categories of controls: management controls, operational controls, technological controls, and data controls.

Managerial Controls

Management controls generally involve the setting and oversight of broad policies, rather than day-to-day actions.

• Top Management Support. At Allfirst and AIB, top management was largely divorced from control issues, leaving them to back-office groups. There was even confusion over whether various control measures should be handled by AIB or Allfirst.

• Risk Assessment. Top management must conduct risk assessments to identify areas of high risk. Certainly, trading large options is highly risky if not properly controlled. In addition, it requires specialized knowledge, and allowing an inexperienced trader to do it in a company like Allfirst, where it was outside of the firm's main line of business, was obviously a high-risk situation that needed extensive control.

• Policy Development. A company needs to establish policies for activities involving high risks. Policies give strong direction to lower-level employees, and they ensure (if followed) that all activities will be treated equally.

• Policy Follow-Up. Creating policies good, but if they are not followed, then they mean nothing. Management must check that its policies are being followed through some sort of auditing function.

• Establishing Independent Audits and Reports. Sarbanes-Oxley requires firms to create audit committees that are independent of the company's senior management. They also require that companies set up a "whistle blower's" hotline for any employee to contact the independent audit committee. This committee's and hot line's scope should extend beyond financial reporting.
Enforcing Sanctions. When employees violate policies, the firm needs to implement sanctions. Otherwise, employees have only limited incentives to follow corporate policies.

**Operational Controls**

- Good Development Practice. There should be controls on application development for all spreadsheets. This should include initial risk assessment, development methods and practices, and testing.

- Cell Protection. One development practice is particularly important. This is creating cell protection to prevent users from overwriting formula cells. This is always a concern for innocent error, and it is even more of a concern for fraud.

- Operational Audits. Creating policies is good, but if they are not followed, then they mean nothing. Management must check that its policies are being followed through an active auditing function.

- Separation of Duties. There should be clear separation of duties so that at least collusion will be necessary for fraud to succeed. For instance, it may make sense to separate the functions of spreadsheet developer and spreadsheet user.

- Controlling Developer Access. In programming, it is common to have development be done on a single machine to which the developer has access. After development, the program should be moved to a testing machine. On the testing machine, testers should have access, but the developer does not. Finally, the program is moved to a production machine on which neither the developer nor testers have access.

- Requiring Testing. There should be required procedures for testing that specify all of the aspects of testing described earlier.

- Training in Paranoid Testing. Code inspection normally looks for innocent errors. Companies need to develop practices to identify fraudulent deception, including attempts to hide these subterfuges.

- General Employee Training. It is important to train all employees to be wise consumers of spreadsheet results. This includes creating healthy skepticism about the trustworthiness of spreadsheets in terms of both errors and deception.

**Technological Controls**

When firms look at spreadsheets, they usually are dismayed by the poor fraud controls built into spreadsheet programs. This is an area that badly needs to be developed by Microsoft and other vendors. The following is a list of possible technological controls that need to be considered.
Change Auditing. With spreadsheets today, it is difficult or impossible to know who made changes to the spreadsheet, what they did, and what the spreadsheet said before. Change auditing needs to be added to spreadsheet programs if fraud is to be controllable.

Usage Auditing. Similarly, when a spreadsheet is used, multiple people may enter data. For the separation of responsibilities during data entry, the spreadsheet program should be able to record who enters which values.

Authentication. Auditing requires strong authentication. Passwords have proven to be relatively easy to guess or steal. For high-risk spreadsheets, stronger authentication methods should be offered by spreadsheet programs.

Compilation Tools. One useful feature in spreadsheets would be the ability to compile spreadsheets. Users would only be able to enter data and would not be able to change the logic. Although cell protection should be able to do this, it is too easy to circumvent.

Audit Detection Tools. In security, when certain audit events occur, intrusion detection systems send warnings to security administrators. Spreadsheet programs need to develop the ability for certain events to be audited during spreadsheet development and use and for warnings to be sent to security administrators when certain events occur.

Data Controls

Data controls focus on how data is handled. Sources should be controlled, each input item should be checked for reasonableness (length, range of valid values, etc.), data input operators should be trained, and so forth.

Conclusion

Although Sarbanes-Oxley has focused attention on spreadsheets in financial reporting, once organizations begin to take a hard look at their spreadsheets, they will almost certainly realize that all large and important spreadsheets need careful control.

We know the most about innocent spreadsheet errors. We know that people make errors in approximately 2% of all unique formula cells even with careful development. We know, as a consequence, that all large spreadsheets will contain many errors after careful development. We know that testing is needed to reduce this error rate to a reasonable level, and we generally know how to test spreadsheets and how to develop them so that they are testable.

However, financial reporting is also concerned with fraud. Unfortunately, spreadsheet programs today offer few useful features for dealing with fraud. Financial reporting systems that use spreadsheet programs almost invariably have many manual operations, and spreadsheet programs offer no tools for event auditing. Until spreadsheet programs develop features to address fraud reduction concerns, organizations will have to cope by developing strong management and operational controls.
References


Dent, A., personal communication with the first author via electronic mail, April 2, 1995.


Lukasik, Todd, CPS, private communication with the first author, August 10, 1999.


PriceWaterhouseCoopers. (2004, July). The Use of Spreadsheets: Considerations for Section 404 of the Sarbanes-Oxley. http://www.pwcglobal.com/extweb/service.nsf/8b9d788097dfoe985365e00733e0ba/cd287e03c0aeb7185256f0807f8caa/$FILE/PwC~y2Spreadsheet4O4Sarbox.12d


Figures

Figure 1: Controls

Figure 2: Testing for Material Financial Spreadsheets

For spreadsheets of material importance used in financial reporting, what percentage does your company test?

<table>
<thead>
<tr>
<th>Category</th>
<th>Percentage</th>
</tr>
</thead>
<tbody>
<tr>
<td>Almost None</td>
<td>24%</td>
</tr>
<tr>
<td>Under 10%</td>
<td>20%</td>
</tr>
<tr>
<td>11% to 25%</td>
<td>12%</td>
</tr>
<tr>
<td>Over 25%</td>
<td>17%</td>
</tr>
<tr>
<td>Nearly All</td>
<td>16%</td>
</tr>
<tr>
<td>Not Applicable</td>
<td>11%</td>
</tr>
<tr>
<td>Total</td>
<td>100%</td>
</tr>
<tr>
<td>Respondents</td>
<td>862</td>
</tr>
</tbody>
</table>
### Figure 3: Audits of Real-World Spreadsheets

<table>
<thead>
<tr>
<th>Authors</th>
<th>Year</th>
<th>Number of SSs Audited</th>
<th>Average Size (Cells)</th>
<th>Percent of SSs with Errors</th>
<th>Cell Error Rate</th>
<th>Comment</th>
</tr>
</thead>
<tbody>
<tr>
<td>Hicks</td>
<td>1995</td>
<td>1</td>
<td>3,856</td>
<td>100%</td>
<td>1.2%</td>
<td>One omission error would have caused an error of more than a billion dollars.</td>
</tr>
<tr>
<td>Coopers &amp; Lybrand</td>
<td>1997</td>
<td>23</td>
<td>More than 150 rows</td>
<td>91%</td>
<td>Off by at least 5%. This amount could indicate</td>
<td></td>
</tr>
<tr>
<td>KPMG</td>
<td>1998</td>
<td>22</td>
<td></td>
<td>91%</td>
<td>Only significant errors that could affect decisions.</td>
<td></td>
</tr>
<tr>
<td>Lukasic</td>
<td>1998</td>
<td>2</td>
<td>2,270 &amp; 7,027</td>
<td>100%</td>
<td>2.2%, 2.5%</td>
<td>In Model 2, the investment's value was overstated by 16%. Quite serious.</td>
</tr>
<tr>
<td>Butler</td>
<td>2000</td>
<td>7</td>
<td></td>
<td>86%</td>
<td>0.4%**</td>
<td>Only errors large enough to require additional tax payments.**</td>
</tr>
<tr>
<td>Clermont, Hamin, &amp; Mittemeier</td>
<td>2002</td>
<td>3</td>
<td></td>
<td>100%</td>
<td>1.3%, 6.7%, 0.1%</td>
<td>Computed on the basis of non-empty cells.</td>
</tr>
<tr>
<td>Interview I</td>
<td>2003</td>
<td>~36 / yr</td>
<td></td>
<td>100%</td>
<td>Approximately 5% had extremely serious errors.</td>
<td></td>
</tr>
<tr>
<td>Interview II</td>
<td>2003</td>
<td>~36 / yr</td>
<td></td>
<td>100%</td>
<td>Approximately 5% had extremely serious errors.</td>
<td></td>
</tr>
<tr>
<td>Lawrence and Lee</td>
<td>2004</td>
<td>30</td>
<td>2,182 unique formulas</td>
<td>100%</td>
<td>6.9%</td>
<td>30 most financially significant SSs audited by Mercer Finance &amp; Risk Consulting in previous year.</td>
</tr>
<tr>
<td>Total</td>
<td></td>
<td>88</td>
<td></td>
<td>94%</td>
<td>5.2%</td>
<td></td>
</tr>
</tbody>
</table>

*In 2003, the first author spoke independently with experienced spreadsheet auditors in two different companies in the United Kingdom, where certain spreadsheets must be audited by law. Each audited about three dozen spreadsheets per year. Both said that they had never seen a major spreadsheet that was free of errors. Both also indicated that about five percent of the spreadsheets they audited have very serious errors that would have had major ramifications had they not been caught. Audits were done by single auditors, so from the research on spreadsheet and software auditing, it is likely that half or few of the errors had been caught. In addition, virtually all of the spreadsheets had standard formats required for their specific legal purposes, so error rates may have been lower than they would be for purpose-built spreadsheet designs.

**The low cell error rate probably reflects the fact that the methodology did not inspect all formulas in the spreadsheet but focused on higher-rank formulas. However, error has a strong random component, so not checking all formulas is likely to miss many errors.

Source: Panko (2005)
Figure 4: The COSO Framework

Monitoring

Control Activities

Risk Assessment

Control Environment

Figure 5: COSO/CobiT Framework

<table>
<thead>
<tr>
<th>Corporate Level</th>
<th>Activity Level</th>
<th>GobiT Objectives</th>
<th>Control Environment</th>
<th>Risk Assess.</th>
<th>Control Activities</th>
<th>Info &amp; Comm</th>
<th>Monitoring</th>
</tr>
</thead>
<tbody>
<tr>
<td>X</td>
<td></td>
<td>Planning and Organization</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
</tr>
<tr>
<td>X</td>
<td></td>
<td>IT strategic plan</td>
<td>X</td>
<td>X</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>X</td>
<td></td>
<td>Information architecture</td>
<td>X</td>
<td>X</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>X</td>
<td></td>
<td>Technological direction</td>
<td>X</td>
<td>X</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>X</td>
<td></td>
<td>IT organization/relationships</td>
<td>X</td>
<td>X</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>X</td>
<td></td>
<td>Manage IT investment</td>
<td>X</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>X</td>
<td></td>
<td>Communication aims &amp; directions</td>
<td>X</td>
<td>X</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>X</td>
<td></td>
<td>Manage human resources</td>
<td>X</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>X</td>
<td></td>
<td>Ensure compliance</td>
<td>X</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>X</td>
<td></td>
<td>Assess risks</td>
<td>X</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>X</td>
<td></td>
<td>Manage projects</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>X</td>
<td></td>
<td>Manage quality</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Acquisition and Implementation</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>--------------------------------</td>
<td>--------------------------------</td>
<td>----------------</td>
<td>----------------</td>
<td>----------------</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>Identify automated solutions</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>X</td>
<td>Acquire/Develop app. software</td>
<td>X</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>X</td>
<td>Acquire technological infrastructure</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>X</td>
<td>Develop &amp; maintain procedures</td>
<td>X</td>
<td>X</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>X</td>
<td>Install and test systems</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>X</td>
<td>Manage changes</td>
<td>X</td>
<td></td>
<td>X</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>Delivery and Support</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>X</td>
<td>Define and manage service levels</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>X</td>
<td>Manage third-party services</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>X</td>
<td>Manage performance and capacity</td>
<td>X</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>Ensure continuous service</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>X</td>
<td>Ensure systems security</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>Identify and allocate costs</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>X</td>
<td>Educate and train users</td>
<td>X</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>Assist and advise customers</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>X</td>
<td>Manage the configuration</td>
<td></td>
<td></td>
<td>X</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>X</td>
<td>Manage problems and incidents</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>X</td>
<td>Manage data</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>X</td>
<td>Manage facilities</td>
<td></td>
<td></td>
<td>X</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>X</td>
<td>Manage operations</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>Monitoring</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>X</td>
<td>Monitor the process</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>X</td>
<td>Assess internal control adequacy</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>X</td>
<td>Independent assurance</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>X</td>
<td>Provide for independent audit</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Figure 6: COSO, CobiT, 17799, Common Criteria, and ITIL

![Diagram showing the relationship between COSO, CobiT, ISO/IEC 17799, ITIL, and Common Criteria]

Figure 7: Testing Methods for Material Financial Spreadsheets

When your firm tests spreadsheets of material importance used in financial reporting, which of the following methods does it use?

- All Testing: 100%
- Does Not Test All Cells: 73%
- Not Applicable: 15%
- Tests All Cells: 12%
  - Single Tester: 49%
  - Multiple Testers: 24%
  - Single Tester: 10%
  - Multiple Testers: 2%

Respondents: 861 Financial Professionals
Figure 8: The Fraud Triangle


Figures

Figure 1: Controls .................................................. 16
Figure 2: Testing for Material Financial Spreadsheets ........................................... 18
Figure 3: Audits of Real-World Spreadsheets ......................................................... 19
Figure 4: The COSO Framework ........................................................................... 26
Figure 5: COSO/CobiT Framework ....................................................................... 28
Figure 6: COSO, CobiT, 17799, Common Criteria, and ITIL .................................. 30
Figure 7: Testing Methods for Material Financial Spreadsheets .............................. 38
Figure 8: The Fraud Triangle .............................................................................. 45
4 Spreadsheet Risk and Corporate Responsibility – are UK/EU Directors Liable?

Pat Cleary, Lynne Norris-Jones, Mukul Madahar
Business School
University of Wales Institute Cardiff
Colchester Avenue
Cardiff, CF23 9XR
pmcleary@uwic.ac.uk

Abstract

This management summary seeks to briefly review some of the UK and EU legislation and regulations in order to provide a background to a simple question:

Do directors of UK and EU organisations bear legal responsibility for the accuracy of spreadsheet models used within their organisations?

Answering this question may help researchers gain collaboration from organisations, concentrate the minds of those bearing the responsibility and prompt further research.

1 INTRODUCTION
Since the first EuSpRIG conference in 2000, researchers at UWIC Business School have been attempting to raise the profile of the risks associated with spreadsheet use among the UK corporate community. While some interest has been generated within the public (mainly government agencies such as HM Customs & Excise and Companies House) and financial sectors, no funding has been generated.

More recently, organisations have been willing to listen to research proposals when compliance issues have been mentioned. Concern with liability under SOX and similar legislation and regulation seems to be concentrating minds, particularly where corporate liability is concerned. It was thought that an examination of liability for spreadsheet accuracy might be useful. The purpose of this management summary is to ask a simple question which has been alluded to in the past but the answer to which has not been clearly stated: do the directors of organisations carry legal responsibility for ensuring that the risks associated with spreadsheet use in their organisations are properly managed? The paper briefly summarises the legislation/regulations before posing the question to the conference forum.
2. A SUMMARY OF RELEVANT LEGISLATION AND REGULATIONS

This is not intended to be a comprehensive list of legislation and regulations in any or all of the regions under consideration.

2.1 The Sarbanes Oxley Act 2002

Introduction

Sarbanes Oxley (SOX) is a US law that was passed in 2002 to strengthen corporate governance (and hence compliance) and to foster investor confidence at a time when investors were becoming disillusioned by a series of US corporate and accounting scandals involving e.g. Enron, Worldcom etc. (Knights, 2004).

The legislation is wide ranging, producing new standards for all public company boards, public accounting and management in general.

The aim of the legislation was to introduce best practice for decision-making at management board level, proper accountability and audit procedures together with the development of positive relationships with shareholders and institutional shareholders.

The US approach to the above is prescriptive whereas the EU and UK have taken more subtle (non compulsory) measures, based upon justification and disclosure (Kelly, 2004).

The Sarbanes Oxley Principles:

1. The legislation introduced new standards for corporate boards and Audit Committees;
2. Corporate Management is now subject to accountability standards and criminal action / penalties for failure to meet those standards;
3. Independent standards for External Auditors;
4. Introduction of a Public Company Accounting Oversight Board (PCOAB) regulated by the Security and Exchange Commission (SEC). This has the role of overseeing public accounting firms and issuing them with relevant standards (Knights, 2004).

Main Provisions of the Act

1. Any public company with more than $75m, closing its 2004 fiscal year on or after 25th November had to comply with S. 404 of the Act within 75 days of the year-end date.
2. S. 404 states that company auditors must identify any ‘material weakness’ or ‘significant deficiency’ in verifying the operational reliability of its compliance report.
3. UK-based subsidiaries of US corporations must also comply with the rules as to the integrity of financial data and report sharing in line with its US partners (ibid.).
UK Requirements

Any company with a listing on a US exchange (e.g. the New York Stock Exchange) must comply with SOX. This also applies to European companies with their headquarters outside the US.

The law applies to storage of electronic records in addition to finances. This means that UK subsidiaries of US corporations must give assurances about the integrity of data in reporting to its US parent (Knights, 2003).

The Sanctions

Breach of SOX carries penalties of up to 20 years' imprisonment for Chief Executive Officers (CEOs) and Chief Finance Officers (CFOs) and fines of up to $5m per breach.

The above sanctions ensure that new levels of compliance are adopted in the ways in which companies store and re-use data (ibid.).

2.2 The UK Response

UK companies want the Government to make business related legislation simpler to understand and are frustrated by the increasing rules and regulations bound up in the legislation applicable to them (Watson, 2004).

In May 2004, the UK's Money Laundering Regulations 2003 introduced regulations for all organisations dealing with large volumes of cash. The Regulations specify that organisations manage their data effectively and implement automated systems with the technological ability to identify problems as they arise, and to find rapid solutions for them (Watson, 2003).

Ease of Compliance

a) Security

Security represents an important part of corporate compliance within both the US Sarbanes Oxley regulation 2002 and the UK Turnbull Report published in 2000. Both look to corporations having the ability to prove that they have complied with the 'spirit' of the regulations. Indeed the Turnbull Report specified the need for organisations to consider operation risks as financial risks and to treat them accordingly (Bradbury, 2003).

b) Data Mining and Reporting

Business intelligence provides organisational support for compliance regulations by designing tools to assist Chief Executives in reporting financial data to shareholders on a more consistent and regular basis (ibid.).
c) IT Governance

IT departments have to build in compliance budgets (both financial and people based) to accommodate compliance projects and to ensure that practical deadlines can be set for completion of compliance rules (ibid.).

The above is all the more important in current day-to-day businesses in which compliance rules have moved to the board room.

A Sample of Regulations for UK and EU Compliance

The Regulations are applicable to banks and financial organisations, in the wake of the September 11th 2001 attack, and are concerned with the manner in which they manage operational risks through risk assessment exercises. SOX, clearly, is also applicable as stated above (Bennett, 2003).

a) Basel II

The Basel Capital Accord requires banks and financial institutions to build into their reporting systems a clear understanding of operational risks. This involves major data management procedures involving an assessment of data storage facilities throughout the organisation. The system is based upon an assessment of 5 years' historical data to ensure compliance (Watson, 2003).

b) International Financial Reporting Standards

In 2003 the EU Council of Ministers agreed a regulation for all Member States to prepare their financial statements via the International Accounting Standards Board (IASB) by 2005 (ibid.).

c) The Higgs Report

This led to a Combined Code on Corporate Governance, applicable only in the UK for companies listed on the London Stock Exchange (Bradbury, 2003).

d) Companies (Audit, Investigation and Community Enterprise) Act 2004

Article 2.10 amends the requirements in relation to the contents of the consolidated annual report so that more detail is required and more analysis of the business risks is included.
3. The Question

The key challenges being faced by UK organisations, as highlighted, are issues related to data ownership and accountability. Organisations need to deliver complete, quality data on a regular basis. There are requirements for disclosure and transparency of data. The above discussion clearly indicates that the way information is stored within organisations is subject to compliance regulations. With significant amounts of financial data being stored in or interpreted through spreadsheets, their development and data stored in them is subject to these regulations.

Suppose an organisation makes a decision, based upon faulty spreadsheet models, which proves to be wrong and causes significant loss to its stakeholders. A number of scenarios could apply:

a. The organisation was not aware of the risks associated with spreadsheet use and had no strategy for risk reduction and no policy for spreadsheet model development.
b. The organisation was aware of the risks associated with spreadsheet use but had taken no action to reduce those risks.
c. The organisation was aware of the risks associated with spreadsheet use and had formulated strategies for risks reduction but these were not enforced.
d. The organisation was aware of the risks associated with spreadsheet use, had formulated and enforced appropriate strategies for risk reduction which had proved ineffective in this particular case.

With reference to the SOX legislation and UK/EU regulations, would any of these scenarios lead to directors of the organisation being legally responsible?

References:


Bradbury, D. (2003) Business and IT need to work together on compliance, Computing (May) www.pcmag.co.uk/features/1150631;


Kelly, L. (2004) Data retention takes priority, Computing (February) www.pcmag.co.uk/features/1152799;


Watson, J. (2004) Legislation increases pressure on business, Computing (March) www.pcmag.co.uk/analysis/1153774;
ABSTRACT

Previous research on spreadsheet risks has predominantly focussed on errors inadvertently introduced by spreadsheet writers i.e. it focussed on the “end-user aspects” of spreadsheet development. When analyzing a faulty spreadsheet, one might not be able to determine whether a particular error (fault) has been made by mistake or with fraudulent intentions. However, the fences protecting against fraudulent errors have to be different from those shielding against inadvertent mistakes.

Faults resulting from errors committed inadvertently can be prevented ab initio by tools that notify the spreadsheet writer about potential problems whereas faults that are introduced on purpose have to be discovered by auditors without the cooperation of their originators. Even worse, some spreadsheet writers will do their best to conceal fraudulent parts of their spreadsheets from auditors. In this paper we survey the available means for fraud protection by contrasting approaches suitable for spreadsheets with those known from fraud protection for conventional software.

1 MOTIVATION

The economic significance of spreadsheet errors is documented by the spectacular cases listed on the EuSpRIG web site [EuSpRIG]. The high frequency of errors of various kinds is documented by a host of empirical studies. An overview of these studies can be found in [Panko 00] or seen on Ray Panko’s web site [Panko]. However, they could not reveal the motives underlying a particular error. Even in the study we did a while ago [Clermont et. al, 02] or in other un-documented assessments of sheets, we cannot state for sure that none of the deficiencies identified had a fraudulent background. We can only state that the way the original spreadsheet developers cooperated lets us assume that the majority of errors – if not all – were introduced without any hidden purpose.

Due to the Sarbanes-Oxley act, managers cannot rely on such well-meaning assumptions. Hence, research has to look deeper into the causes and consequences of spreadsheet errors. In this context it is important to note that risk prevention strategies always have to start at the head of the chain of problems to identify proper mechanisms to prevent them. For example, assuming that the kind of problems spreadsheet writers are dealing with remains of the same nature, investing in a person’s spreadsheet specific education will
hopefully have a positive impact on the error rates of the sheets this person writes\(^1\). This investment will, however, be without any effect, or possibly even have negative effects, if this particular employee is introducing errors on purpose.

To survey mechanisms protecting against the fraudulent introduction of faults or the fraudulent modifications of spreadsheets, one might review the general literature on software fraud considering the differences between spreadsheets and software developed by state of the art third generation languages and their respective professional development processes [Baskerville 93].

In the following sections we first consider general software crime and related protection mechanisms as well as the specific distinguishing features of spreadsheets. Based on these considerations, we will look at fraud protection mechanisms for spreadsheets. Unfortunately, we have to conclude that instantiating a comprehensive strategy still requires actions by both vendors of spreadsheet software and by the respective user organizations. Last but not least, it requires further research.

2 SOFTWARE CRIME

2.1 Strategies Dealing with Conventional Software

With conventional software systems, the distinction between program and data is clearly defined. In its weakest form, we have the compilation process as a universal frontier between software development, or its modification during maintenance, and the use of this software by having it operate on some data. Normally, users are distinct from developers and usually, software development is even further decoupled from its use, e.g. by distinguishing between development machines and production computers. Organizational precautions will ensure that there is only a one-way street from development to operation.

This distinction allows for a clear separation of the strategies protecting against frauds due to data manipulation and frauds due to software manipulation per se. Thus, frauds on the data level can be addressed by specific access rights. While fraudulent data manipulations can be made more difficult by the specific design of the data base schema or by incorporating cross checks in the application software, such strategies would have only a reduced effect were there to be no separation between developers and users.

The main strategy for assuring that software performs to its specification and only to its specification lies in the general quality assurance mechanisms of reviews and in testing. The advantage of reviews is that they can be made early in the development process. For fraud protection they will normally be executed in the form of inspections with item-lists specifically geared towards the identification of hidden functionality. The specifics of such lists are, as usual with inspections, language specific.

The systematic approach inherent in anti-fraud inspections has also to be used for anti-fraud testing. Among the vast spectrum of testing methods proposed, see [Zhu 97], one would specifically select functional testing methodologies. However, the problem is that one ought to test against the presence of code that is not required in the specification or – assuming the design has been adequately reviewed – not foreseen in the design of this

\(^1\) This positive assumption seems invalidated by [Panko, Sprague 97] claiming that the error rate is invariable with respect to seniority of the spreadsheet developers. However, this data is based on tasks of different complexity. Here, we assume risk specific education.
particular software. To account for this, mutation testing [Murnane, Reed 01] might be an adequate approach.

Finally, the interaction of data and software might also be used. Placing specific intermediate assertions will help to ensure that even intermediate results have to remain in correspondence with the conditions asserted. They are important in the fraud prevention context because persons operating with the system cannot use clever tricks – an alternative they might have even in the presence of openly visible checksums – to circumvent them. Assertions are embedded in code and therefore hidden from the user.

An aspect to be considered in conventional software that is less relevant for spreadsheets is that of hardware related frauds. The assumption is that spreadsheet users, be they also developers or only parametric users, have the sheet on their own PC. Hence, except for special cases, we can see almost no room for the use of encryption of data. Privacy aspects dealt with in the literature on security, specifically on the literature on database security [Pernul et.al. 98] are relevant in general though. They are relevant concerning source data used in spreadsheet computations and in the transmission of spreadsheet results if those are to be used by persons other than the person operating a sheet.

2.2 Spreadsheet Specifics

Before checking to which extent the strategy “Let’s do for spreadsheets what serves well for conventional software!” is used, one has to consider the differences between general software and spreadsheets.

A basic difference usually dwelt upon is that conventional software is produced by software professionals and spreadsheet software is developed by end users. Embarking too much on this difference would be dangerous though. It invites one equating end users with people of limited technical capability. This assumption could be quite misleading. Though spreadsheets can be written without too much in-depth knowledge of either programming or modelling, those that are subject to fraud are definitely not written by novices. One should rather assume that they are developed by personnel who, though being classified as end users, are well versed in both the domain they are working in and in spreadsheets themselves.

Spreadsheet writers differ from professional software developers, though, by following another development process. While 3GL software is usually created by following a rather structured development process based on some kind of specification, spreadsheets are usually written in a much more explorative fashion [Nardi, Miller, 1990].

This might be a consequence of the fact that, in contrast to spreadsheet development where developers unite all three roles, in the case of conventional software, the requirements owner and programmer are different individuals. Thus, even with relatively unstructured agile processes, specific steps have to be followed to allow for the coordinated effort of several people. Furthermore, even if the process is free from formal specifications, there are documents – whether in form of story cards [Beck 05] or in more conventional form is of secondary importance only – that can serve as reference points against which particular pieces of the system can be verified. With end user programming, i.e. whenever the requirements owner and the developer collapse into a single person, such reference documents seem to become unnecessary. Hence they exist only rarely. Even if the spreadsheet writer develops an explicit, written model of what is to be casted into the computations performed by the sheet and even if an independent design of the sheet has been made, these documents usually do not have the rigor of professional documentation.
Therefore, while the development of classical software will produce some *upstream documents* that can serve as reference points for the final code, this option is not available in spreadsheet development. Thus, the fact that the spreadsheet is written by a domain expert has severe direct and indirect consequences for fraud prevention strategies.

Besides these process-related aspects, one has to consider the *tool* itself. 3GL software is written with the support of some development tool, an editor that might be arbitrarily independent of the language used. The formal text (source code) thus written is transformed in an automated step (compilation) to executable code. Only then is the compiled code applied to the data it is assumed to transform or to process. Spreadsheets, on the contrary, offer a 2(+1)-dimensional arrangement of cells such that each cell can be either empty, contain a constant value, or a formula. Constant values can be considered to be either part of the program, serving as constants or labels, or as data to be processed, which would be called ‘input’ if we were discussing conventional, 3GL programs.

The analogy of output is missing insofar as each cell that has some contents displays its value, irrespective of whether this value is to be assumed as result of the spreadsheet model or whether it is just an intermediate. Methodologies prescribe modularizing sheets such that distinct input-, computation-, and output areas appear on the sheet [Ronen et al., 89], [Burnett, 01]. However, these methodologies are observed only to a limited extent and one may also raise arguments against their adoption.

This *coupling of program with data* has, however, severe consequences for quality assurance and authentication procedures. Certainly, one could test spreadsheets in the same way as conventional programs are tested. In the conventional case, however, the tester is free from the risk of inadvertently changing the program; with spreadsheets, this risk can only be controlled by cell protection and/or locking mechanisms [Burnett et. al., 01].

### 3 PROTECTION MECHANISMS FOR SPREADSHEETS

In this section we discuss mechanisms for protecting spreadsheets against errors introduced in a fraudulent manner. Before delving deeper into this issue, it is important to say that fraud is a socio-psychological phenomenon with economic consequences. Hence, as with other social or psychological phenomena, technical means alone are insufficient and technical or methodological means cannot prevent fraud. They can only establish barriers that are difficult to overcome. Ingenious people, however, will always be smart enough either to overcome or to circumvent such barriers. Hence, *organizational means* are necessary to embed the recommendations discussed below in an appropriate, organization-specific manner to avoid having malicious users being able to circumvent what is placed in front of them to protect the integrity of models and computations.

Having said so, we discuss in the following sections how reviewing, testing, the placement of assertions, and authentication – techniques taken from general software fraud protection – can be used to protect spreadsheets against the incorporation of fraud-permissive code or against fraudulent modifications.

### 3.1 Reviews and Inspections

In conventional software development, inspections, though still not universally used in practice, are attributed to be the most effective error detection process steps [Endres, Rombach 03, Rombach et.al., 02]. Interestingly, not all software in use has been subjected
to a review while nobody would trust an untested piece of software. The fact that people trust more in the less strong testing technology than in reviews should be an important point to consider when devising a comprehensive fault prevention strategy. It also constitutes an interesting research question in its own right.

There are several forms of reviews, distinguished by their purpose, position in the development cycle and the particular techniques applied. The techniques range from walkthroughs to inspections. While walkthroughs amount essentially to a manual execution of the programme in a particular group setting, inspections usually check code linearly against specific deficiencies. The deficiencies are taken from a list that contains items that are known to be particularly error prone within the given development context. However, one might follow different reading strategies [Biffl, 01]. Reviewing development products from pre-coding phases normally requires a specific reading strategy that takes care of the non-linearity of the contents of the respective document. In any case, one has the requirement that the full document is checked and the reading strategy has to ensure that nothing is missed.

When checking spreadsheets, one has first to note that they do not share the linearity of 3GL-programs. The arrangement of cells is two-dimensional on the sheet and if there are references between different sheets of a work-book, one might consider a spreadsheet as three-dimensional arrangement of interrelated cells. Hence, particular recommendations for a reading strategy are needed. Of course, one might follow a line-by-line, column-by-column strategy. This will ensure that each cell is checked, a requirement called for by most recommendations [Rothermel, 01; Sajaniemi, 98; Burnett, 01]. However, one might doubt whether the high number of cells present in professional spreadsheets makes this strategy productive. Cell references can stretch over arbitrary distances and hence this pseudo-linear reading might miss important semantic relationships, relationships that were placed (or missed) with fraudulent intentions.

Further, the fact that many spreadsheets contain substantial portions of a repetitive nature must be considered. If ignored, reviewers may become bored with a resulting loss in care and attention. However, becoming bored or tired is a problem for any kind of reviews. Therefore, irrespective of the special review strategies, reviewing methodologies call for strict time limits to be observed in review sessions.

The work of the Klagenfurt group proposes strategies to lessen this effect. In [Ayalew et al., 00; Clermont 03], logical areas are proposed. They group cells of identical content into sets that can then be checked as sets on their defining basis. Originally, this concept was developed to identify differences between regularity in the layout and regularity in the semantic content of a cell. An empirical evaluation [Clermont et. al, 02] has shown that the related tool for supporting a reviewer not only has a high detective power – even in sheets that were considered correct, the reviewer found error rates that are in line with those reported in the literature [Panko, 98] – it also dramatically lessens the time needed for performing the inspections.

This effect is to be expected by comparing the approach to other software certification strategies. In conventional software development, (full) path testing is accepted as the most thorough coverage strategy. In practice, full path testing is infeasible though [Weyuker 86]. Hence, one embarks on less rigorous strategies such as simple path tests where for each loop at least one case that executes the body and one case that misses the loop body are run.
With spreadsheets, we assume that the analogues of loops are cells with content copied from other cells. The finiteness of the (2+1) dimensions of the sheet would allow an exhaustive assessment of each cell. However, performing just a check of each logical area amounts exactly to what is checked with a simple path test. Moreover, since the areas are identified automatically, even minor differences in arbitrarily complex cell contents will be identified. Thus, if fraudulent modifications lead to irregularity in structures, reviewers are specifically pointed to such spots. If fraudulent modifications lead to consistent modification, the reduced number of items to be considered improves the chance that they do not slip the attention of the reviewers.

The concept of grouping repetitive structures for review purposes can be extended to groups of cells that are defined according to a repetitive pattern. With the so defined semantic classes [Mittermeir, Clermont 02], even higher levels of aggregation can be safely checked for fraudulent deviations.

These concepts have been formally defined in [Clermont, 03] where an orthogonal concept based on the data flow between semantic units is additionally defined. With this, a reviewing discipline is proposed that is (or strives to be) in accordance with the application model formulated by the spreadsheet. How the concepts just described can be combined for auditing has been described in [Clermont, Mittermeir, 03]. There, three specific reading strategies for spreadsheets are proposed.

It should be mentioned here that in searching for fraud, several rounds of passing through a sheet are necessary and these rounds might follow different sheet traversal strategies [Biffl, Halling, 03]. While a check for accessing files that do not belong to the legitimate interface might be done automatically by a line-wise or column-wise check of each cell (including hidden cells!), auditing the algorithm itself might instead require a trace along or against the flow of data. A rather selective strategy might be to check for cases where a choice of misleading labels can lead to effects that are to be prevented.

3.2. Testing

Research on testing conventional software has produced a host of literature (see [Zhu 97, Hierons 02] for an overview). Rothermel and his co-authors proposed a specific testing methodology for spreadsheets [Rothermel 01].

Given Dijkstra’s statement that testing can only show the presence of faults, but never their absence [Dijkstra 72], the high emphasis on testing might be surprising. Dijkstra’s remark applies in an analogous manner to fraudulent deviations of a program from what should be the program following its specification. In fact, testing is weaker than either proofs or checking against specifications in a review. The reason for this is that the latter two deal with a program in its intentional form while testing deals only with a sample taken from the possible extensions of input-output transformations to be performed by a program. Hence, while testing can only raise the level of confidence one can have in the correctness of a program, its fault detection characteristics are sufficiently different from those of reviews or proofs so that any solid quality management strategy would employ a mix of these quality assurance strategies [Mittermeir, 04].

A basic assumption of testing is that it is performed after coding is completed. In industrial software production, the separation of the coding and testing phases is even emphasised by having different personnel for programming and for testing and clear prescriptions on the quality a module has to have before it is to be passed to the testing group and subjected to formal testing. Configuration management systems and special
conventions are needed to ensure that there is no conflict of versions between testing and correcting.

Due to the integration of data and code, such assumptions seem problematic with spreadsheets. Any change of (input-) data might, by mistake, lead to a change in the program itself. Further, manual instrumentation of (normal) spreadsheets seems impossible, since such changes would distort not only the internal structure of the sheet, but also its external appearance. Cell protection might be an answer to the risk of inadvertently changing a cell that belongs to the “program part” rather than to the “input part” of the sheet. Doing so on a discretionary basis might itself be error prone if one aims at testing to raise the confidence in the functional correctness of the sheet. If one plans to use testing as fraud identification strategy, however, it would appear to be totally insufficient.

To improve confidence in testing as a fraud identification strategy, two changes seem mandatory:

a) the physical separation of data from code,

b) the temporal separation of data changes from changes in code.

The first requirement is approached by having methods ensuring firstly specific regions of the sheet for each one of the three categories: input cells, “code”, and cells bearing final results. Secondly, block-wise protection locks are needed for the latter two categories of cells. To provide separation with a higher level of protection from the system would, however, require a methodology where workbooks are structured in data-sheets and program sheets such that the front sheet shows only input values and the final result values. Computations would then take place on a different sheet (or possibly on a stack of different sheets) of the workbook.

This separation will require some discipline from the developers, but it does not really raise the overall complexity. In contrast to other proposals that recommend separation of formulas from input cells, here the lower level sheets can be prepared in exactly the manner in which they would have been prepared anyway. Even the initial desk checks might be done by the original spreadsheet writer as she or he used to do. Only before submitting the sheet to testing, all values in intended “input cells” have to be replaced by a reference to the respective cell on the front sheet and, for all “output cells”, the front sheet has to get a simple reference to the respective “program sheet”. Labelling information still has to be copied to the “front sheet” containing the value view of this spreadsheet model. This could, however, be done with tool support if one follows labelling identification strategies such as proposed by [Hipfl, 04].

However, in the fraud prevention case, the crucial issue remains whether the sheet that has been tested will also be the sheet that is actually used in operation. To achieve this, one would need to totally and finally remove the code modification capability from the person operating the sheet. The OpenOffice extension TellTable [Nash et al, 04] supports this strategy by placing the real spreadsheet on a protected server. Users operate on a view of the sheet presented via a VNC client. Thus, they are allowed to perform parametric variation of the sheet while the algorithmic core is protected and can be changed only by a special group of users. This separation of interface from internals of the sheet would also enhance auditability of change traces. With tools supporting focussed analysis of modification histories [Baxter, 04], auditors can efficiently ensure that the sheets in operation are still conformant to those that have been certified previously.
Another answer to this issue would be to use compiled spreadsheets. Compiled spreadsheets seem to be not really an additional burden on the spreadsheet development process. The spreadsheet compiler would just clearly define the boundary between development and operation. In fact, the compiled spreadsheet would, from the user’s point of view, just consist of the “front sheet” proposed above. However, this front sheet is not manually constructed but prepared by the spreadsheet compiler. Underneath this front sheet, there can be arbitrarily optimized code that might be protected in a highly professional manner.

It is to be seen that the advantage of using a spreadsheet compiler over the use of protected cells might be limited for the initial version of a spreadsheet. However, those sheets that are fraud-critical usually have a long lifespan during which they get modified. With the compiler solution, a trace history over the various versions of such sheets has the compilation run as a clearly defined event that marks the creation of a new version.

A word of caution still has to be said. As with searching for errors, searching for frauds is exploring the open universe against the finite space of the specified (or intended) solution. However, in seeking errors, one has a rich empirical basis for the kinds of errors humans commit. In the case of seeking for fraudulent deviations from the specified (or intended) solution, this experience base is very limited. Hence, the identification of appropriate test data sets, which basically amounts to a very carefully designed sampling strategy, is much more difficult and laden with risk.

3.3. Assertions

Assertions are a strategy for relating program code to its specification and, if the assertions are to be executed during operation, for checking whether the state defined by the operation of the program on a particular stream of input data conforms to the specification.

With spreadsheets, assertions play a limited role if one is not prepared to accept checksums as a weak proxy of such assertions. However, we see in the work of Ayalew [Ayalew 01, Ayalew et al., 00] a certain approximation to assertions that might have some merit in fraud prevention. His idea was to have a layer underneath the normal spreadsheet on which the ranges to be assumed by the values of selected cells can be specified. Using interval arithmetic, the specified ranges can then be compared with the ranges resulting from executing the functions and operations of the original spreadsheets on the ranges of cells higher up in the data flow. This allows establishing inclusion relationships between the actual value, the computed range, and the expected range of result cells. Deviations will point to potential faults in either the spreadsheet writer’s model or in its implementation in the original spreadsheet.

While the direct transfer of this idea to fraud prevention might be of limited use, its generalization seems highly promising. It offers a layer underneath the spreadsheet that is under the application expert’s control and that checks certain spots of the sheet in a way that is not directly accessible to the application expert operating the sheet. Thus, modifications of the original sheet will be possible as long as these modifications remain within the competencies granted to this person. Obviously, this also requires additional protection mechanisms as discussed in connection with a spreadsheet compiler.

3.4. Authentification

Finally, one has to point to classical security mechanisms such as authentification or water marking ([Craver et al., 98], [Collberg et. al., 99]). Both rest on cryptographic tech-
niques and would, to be of practical use, require compiled sheets or at least a strong physical separation between data and code.

The idea is basically that the content of the program portion of the sheet is subjected to some cryptographic encoding such that any modification in the program would lead to a different cipher. Consequently, the program is sealed against unauthorised changes and users of this program can be sure that they used the program they intended to use and not some slightly modified clone of it.

Since adequate treatment of this topic would extend the scope of this paper, we refer to the related literature for cryptographic protection of conventional software (c.f. [Devanbu, Stubblebine, 00]). Here, the differences between conventional software and spreadsheets will not matter as long as one can clearly separate code from data. It is to be understood, however, that applying any of the techniques discussed above, regardless of how efficient or effective they might be for dealing with inadvertent errors, will be wasted effort as long as one cannot be sure that the sheet that was operated was the sheet that has been previously examined.

4. SUMMARY

Though spreadsheets have to be understood as programs written in a very high level programming language, mechanisms to protect them against fraud have to recognize the substantial differences between them and classical software. Key to these differences is that the multi-person process, which is inherent in conventional software production and has a clear differentiation of roles, is lost in favour of a single person process. The latter is apparently much harder to control from the outside. Intermediate documents that might otherwise serve as reference points are usually missing. To compensate for this, organizational measures have to be put in place. Likewise, one has to take care of the fact that spreadsheet systems work in an interpretative mode and that data and program is not necessarily separated on spreadsheets.

Separation of data and “program” will eventually be a key requirement when one wants to protect spreadsheets against fraudulent modifications, but we must be aware that this would radically change the way spreadsheets are used and developed. Up to now, the approaches suggesting separating input data from programs proper have, in practice, not been blessed with high acceptance.

Inspections of various sorts might help to check sheets. Techniques that have analogous effects to loop aggregation might improve the efficiency and even the effectiveness of such reviews and testing approaches will also have their place in fraud prevention. However, all effort invested will be futile as long as one cannot be sure that the spreadsheet assessed will be the spreadsheet executed. This assurance can be given by cryptographic methods. However, they require a clear identification of what is code and what is data.

5. REFERENCES

[Ayalew, et al 00]
[Ayalew 01]

[Baskerville 93]

[Baxter 04]

[Beck 05]

[Biffl, 01]

[Biffl, Halling, 03]

[Burnett, 01]

[Burnett et. al., 01]

[Clermont, 03]

[Clermont et.al., 02]

[Clermont, Mittermeir, 03]
[Collberg et.al. 99]

[Craver, et al. 98]

[Devanbu, Stubblebine, 00]

[Dijkstra 72]

[Endres, Rombach 03]

[EuSpRIG]

[Hierons 02]

[Hipfl 04]

[Mittermeir 04]

[Mittermeir, Clermont, 02]

[Murnane, Reed 01]

[Nash et al 04]


Acknowledgement
The authors would like to thank Dr. Tom Arbuckle and the anonymous referees for helpful comments on a draft of this paper.
The Importance and Criticality of Spreadsheets in the City of London

Grenville J. Croll
EuSpRIG – European Spreadsheet Risks Interest Group
mail@grenvillecroll.com

Spreadsheets have been with us in their present form for over a quarter of a century. We have become so used to them that we forget that we are using them at all. It may serve us well to stand back for a moment to review where, when and how we use spreadsheets in the financial markets and elsewhere in order to inform research that may guide their future development. In this article I bring together the experiences of a number of senior practitioners who have spent much of their careers working with large spreadsheets that have been and continue to be used to support major financial transactions and manage large institutions in the City of London. The author suggests that the City of London is presently exposed to significant reputational risk through the continued uncontrolled use of critical spreadsheets in the financial markets and elsewhere.

1 INTRODUCTION

The purpose of this paper is to give an overview of the uses to which spreadsheets are put in large organisations and major businesses operating within the City of London (an ancient yet well defined socio-economic area now including Canary Wharf). The aim is to make as explicit as is possible within the bounds of commercial confidentiality the reliance that is placed upon spreadsheets within one of the world’s largest financial markets. The sources of the material are professionals from a variety of large organisations who have kindly granted the author telephone interviews under assurances of anonymity in order to permit a full and frank disclosure. Interviewees consisted of approximately twenty auditors, accountants, actuaries, bankers, directors, insurers, lawyers, quantitative finance specialists, IT specialists, regulators and other individuals playing significant roles in the City of London. The duration of each telephone interview (in each case conducted by this author) was approximately half an hour. The interviews were not recorded, however, hand written notes were taken. The research methodology was inspired by MacMillan [MacMillan, 2000]. Interviewees were called a few days after having received the title, abstract and most of this section by way of introduction. Further material was obtained from an internet discussion forum popular within the financial markets [Wilmott, 2005]. The author has collated and edited the contributions and provides a summary and conclusion.

Interviewees were informed that it was intended that each individual should characterise the uses of spreadsheets in their sector in terms of the people involved and their roles, the number of spreadsheets involved, frequency of use, their size, importance, significance or criticality. Significance or criticality could relate to finance, business opportunity, business risk, commercial necessity, fiduciary or legal duty, individual careers, health & safety etc. It was indicated that information on the professionalism with which spreadsheets are put together, perhaps in terms of Grossman’s eight principles of Spreadsheet Engineering [Grossman, 2002] would be useful. Interviewees were informed that it was hoped that the information gathered from the exercise would complement the formal survey approach presently being undertaken by academic colleagues.
2 THE CITY OF LONDON

The Corporation of London on their website [CofL, 2005] introduce the City of London thus:

“The City of London is the world’s leading international financial and business centre…This sector made a net contribution to the UK's current account of over £13bn, a significant amount of which was generated within the Square Mile. The City contributes 3% to the UK’s GDP”

The City of London’s contribution to UK GDP pales besides its international financial significance [CofL, 2005]:

- $504bn foreign exchange turnover each day in London;
- 45% of the global foreign equity market
- 70% of Eurobonds traded in London
- $2,000bn per annum traded on metals in London
- London is the world's leading market for international insurance. UK worldwide premium income reached £161bn in 2002
- $884bn a day traded on the London international futures exchange
- £2,619bn total assets under management in the UK in 2002
- £1,046bn in overseas earnings generated by the maritime industry in London
- 287 foreign banks in London
- 19% of international bank lending arranged in the UK (largest single market)
- 381 foreign companies listed on the London Stock Exchange (LSE)
- £1,200bn pension fund assets under management (third largest in the world)
- $275bn daily turnover in 'over the counter' derivatives (36% of global share)

The main sectors given are Accountancy, Banking, Business information sources, Exchanges, Finance, Fund Management, Government, Insurance, Management Consulting, Maritime, Public Private Partnerships, Regulation, Transnational organisations and Transport. According to a recent survey [CEBR, 2005] commissioned by the Corporation of London, the City of London supports approximately 317,000 “City-Type” financial jobs. It would be safe to assume that all of these jobs required some familiarity with spreadsheets, most likely Microsoft Excel. We estimate that up to 30% of these jobs would be working with Important, Key or Critical spreadsheets as defined below.

3 DEFINING IMPORTANCE AND CRITICALITY

We originally defined importance in terms of the explicit financial value represented in or by the spreadsheet. An important spreadsheet was one that had lots of large financial numbers in it, where material error in those numbers could cause the organisation significant financial loss of the kind that has been well documented by Panko [Panko, 2000] et al.

A regulator, however, was kind enough to outline the mechanism by which they intend in future to measure the importance or criticality of spreadsheets submitted to them by regulated firms. Generalising their scale for wider use, it would seem sensible to categorise spreadsheets as follows:
Type | Description and Potential Impact of errors
--- | ---
Critical | Material error could compromise a government, a regulator, a financial market, or other significant public entity and cause a breach of the law and/or individual or collective fiduciary duty. May place those responsible at significant risk of criminal and/or civil legal proceedings and/or disciplinary action.
Key | Material error could cause significant business impact in terms of incorrectly stated assets, liabilities, costs, revenues, profits or taxation etc. May place those responsible at risk of adverse publicity and at risk of civil proceedings for negligence or breach of duty and/or internal disciplinary action,
Important | Material error could cause significant impact on the individual in terms of job performance and career progression without directly, greatly, immediately or irreversibly affecting business or the organization.
Store & Retrieve | Spreadsheets used as databases, with few issues other than data correctness and information security and where the impact of error is low.
Expired | Spreadsheets over three years old no longer required in the active management of the business, but may be required to be archived by statute or good practice. Present impact of error is low.
Personal | Other spreadsheets used by the individual in the day to day performance of their duties, where the impact of material error is low.

It was this regulator’s intention that spreadsheets be annotated as above and with a similar confidentiality classification. There was a requirement for Critical spreadsheets that at least two people each side of the regulatory divide were aware of their function & purpose.

The above scale enables us to discuss more precisely the role of spreadsheets in the City of London.

4 THE ROLE OF SPREADSHEETS IN THE CITY OF LONDON

Three differing contributors to an internet forum [Wilmott, 2005] recently addressed the author’s question regarding the present role of spreadsheets in the financial markets:

“Put simply and succinctly, despite the higher operational risk, Excel is everywhere - it is the primary front-line tool of analysis in the financial business. Most traders price deals in spreadsheets and enter them in large-scale deal capture systems afterwards.”
“Excel is utterly pervasive. Nothing large (good or bad) happens without it passing at some time though Excel”

“. .. if a customer wants to do a bespoke trade that cannot be handled in our designated booking system then we have to book it in a spreadsheet and add a few bips to cover operational risk capital charge (and additional maintenance). How many bips to add is subjective”

In a later telephone conversation, a regulator summarised thus:

“Spreadsheets are integral to the function and operation of the global financial system”

When asked to comment upon the previous comment, a senior investment research professional working in a first tier investment bank commented:

“Agreed. In addition I would say that the majority of people who use the financial system are not appreciative of the role that spreadsheets play”

Using the above as an introduction to the central role that spreadsheets play, it would seem pertinent to document in more detail their use in the main market sectors of the City of London. The contributions received from interviewees fell into the following categories: Professional Services, Law, Government, Financial Trading, Regulation, Investment Research, Investment Banking, Electricity & Gas, Rail, Private Finance, Fund Management, Insurance & Re-Insurance and Maritime. Although interviewees covered a broad spectrum, the following sections summarize a few major sectors, outlining the major issues.

4.1 Professional services & Large corporate

The four large professional services firms based in the City of London support the statutory reporting and financial management of most of the UK’s larger companies. A larger number of smaller firms support the remaining large corporates and the next, smaller, tier of commerce. The firms have a substantial overseas clientele.

There is extensive use of spreadsheet based financial and strategic modelling within all professional services firms largely for and on behalf of their large corporate and institutional clients. Controls are in place to ensure that spreadsheets developed for clients are of good quality [Read, N. & Batson, J., 1999] and strategies are in place to manage errors [PWC, 2004]. An earlier paper [Croll, 2002] documents the typical steps involved in determining the correctness of models within the professional services environment. The total number of Key and Critical spreadsheets formally and independently checked for correctness by the professional service firms in the City of London does not presently exceed 1,000 on an annual basis. Many spreadsheets are Key or Critical:

“It is not unusual for millions of pounds of time investment to be reflected in a spreadsheet upon which critical reliance is placed”

The independently checked Key & Critical spreadsheets support the types of transaction that are typically reported in the business sections of the quality daily newspapers and
other business press and thus form a relatively small proportion of the total business transacted in the City of London.

Note that whereas the provisions of the Sarbanes Oxley act affecting US companies, are already in force, Basel II requirements affecting G10 banks and the 8th European Directive affecting listed European Corporates only start to come into effect in 2006.

Spreadsheets are not only methods of controlling operational risk (a key pillar of Basel II) but also are themselves a source of operational risk. Under Basel II, effective operational risk controls mean reduced regulatory capital, ineffective controls mean increased capital. Thus the effectiveness, accuracy and riskiness of spreadsheets is a matter the regulator should (and indeed does) take into account when setting regulatory capital.

Key and Critical spreadsheets developed internally by their large corporate clients do not appear to be subject to the same scrutiny as spreadsheets developed internally by the professional services firms themselves. Whilst regulators and insurers have indicated that large spreadsheets are in use in the management of large financial institutions and corporations, there is no evidence to suggest that any of them are subject to any form of quality control. A regulator has indicated that subject to approval of the criticality scale outlined earlier by the regulator’s main board, Critical spreadsheets in that regulators industry will in future have to go through the full software development lifecycle.

A concern was expressed that during the consummation of a financial transaction, there was often an undue reliance placed upon the spreadsheet commissioned by the banks or the project sponsor which the professional services firm was building or auditing. There was no “independence of mind” if a single spreadsheet was being used, even if all the formulae had been checked as correct. This was reminiscent of some earlier work on the issues of interpretation [Banks, D. & Monday, A., 2002] and overconfidence [Panko, 2003] in the use of spreadsheets. Conversely, it was noted that sometimes spreadsheets did not form a major part of a decision, particularly where considerations such as political strategy or health and safety were also important.

It became clear that in certain industrial sectors, particularly Oil and Gas, the demographic profile of some of the key participants was such that certain knowledge, residing within complex spreadsheets was in danger of being lost as these people retired. A similar comment was aired within the fund management industry, where there is an active and competitive job market and people rapidly move on.

There is evidence that the use of strategic and financial modelling seen within the larger corporate is trickling down into smaller firms. Even in smaller transactions, spreadsheets were seen to be critical as they often implement or crystallise financial or operational benchmarks which post deal could precipitate default in the event of them not being set correctly.

4.2 Financial Trading & Fund Management

There is extensive use of spreadsheets within financial trading of all types. Spreadsheet models used in financial trading are often large and complex. Some models include monte-carlo simulation and optimization routines contained either explicitly within the spreadsheet or implicitly as part of a macro or function. Interviewee comments such as the following were typical:

“The whole industry is run on a spreadsheet”
“Critical – couldn’t run the bank without them”

“Spreadsheets are the user interface into the deal”

Spreadsheets are used to value financial instruments of all types and guide decisions on what to trade and when. They are the prime data source when details of a transaction are uploaded into an institution’s main settlement, risk management or other financial systems. With certain types of specialist or complex transaction, the spreadsheet will remain the prime record.

“This has legal consequences, e.g. for compulsory recordkeeping under Companies legislation or FSA regulation, use of these records as evidence for proving disputed transactions, and the possibility of being required by a court to produce (and authenticate) such records after many years (in some cases, even substantially more than 6 years after their creation)”. [Reed, 2005]

Statutory financial information, including capital adequacy, is communicated to regulators and insurers using large and complex spreadsheets.

Spreadsheets are particularly heavily used in the more innovative and hence more recent parts of the market, particularly credit derivatives, an area of the market with a significant daily turnover. The global market in Credit Derivatives was expected to have reached $4.8 trillion by 2004 [BBA, 2005], with the City of London remaining the world's major centre with over half the global market share. Unsurprisingly, there is a “massive dependence” on the use of spreadsheets in interest rate and equity derivatives.

There is an active software vendor sector providing the financial markets with highly specialist pre-built or custom built spreadsheet based financial functions. This market was first established in the mid eighties, with one vendor now claiming 30,000 clients [MBRM, (2005)].

There is clear evidence that market participants are controlled at a strategic level by senior personnel using spreadsheet models. Errors were experienced and corrected within these strategic spreadsheet models, but no systematic error discovery processes took place.

With the exception of quantitative finance professionals working as software engineers, there is almost no spreadsheet software quality assurance or appreciation of the software development life cycle as it might relate to spreadsheets. Spreadsheets built using well engineered code libraries were inevitably tinkered with later by traders, sales people, analysts and other users in an uncontrolled fashion.

There is endemic denial of the extent of the use of spreadsheets within the financial markets at the highest level even though spreadsheets are very clearly, indeed obviously, being used to support large parts of corporate activity. The ease with which malfeasance and malpractice can occur in conjunction with the use of spreadsheets, and the risks relating thereto, was almost palpable.

“The known weakness in spreadsheet security has already led to some public domain incidents, notably the AIB affair”
A managing director of a regulator (who was not an interviewee) is certainly aware of the risks of end-user computing:

"Are integrated systems a holy grail? Not necessarily. The PC triggered a revolution in financial services products. New complex products are very much built on PC computing capacity and could not have happened without it. And yet, senior management in many financial services firms regard the use of PCs as a temporary solution that needs at some point to be integrated – and somehow larger systems are the only solution. What this thinking demonstrates is not really a concern for productivity but rather system management, control and audit. This “large systems” vision of the world poses risks because as long as systems are run off temporary PC solutions and are not part of larger systems, all of the controls that are in place for larger systems are forgotten about when applied to smaller systems. It is important that companies apply many of the same controls and management over their fragmented systems as they apply to mainframe systems instead of concentrating efforts on migrating local computing power into a single managed system."

An interviewee in a first tier bank reported that there was increasing interaction with corporate lawyers regarding spreadsheets and an increasing trend towards complexity, but there was an increasing trend in the last few years “towards better engineering within large complex spreadsheets”. Whether the latter comment is applicable in less tightly regulated sectors such as Hedge Funds remains to be seen.

Spreadsheets also perform a critical role in the fund management industry. An actuary working in a very large fund reported that 90% of their working day was spent within the spreadsheet environment, and that this was typical. The daily profit and loss and balance sheet for a fund, performance measurement, a “great deal” of analysis, asset liability matching, portfolio optimisation (eg of a £20Bn fund) and extensive data manipulation all occur within the spreadsheet environment. Spreadsheets were felt to be indispensable for anything innovative.

There is evidence that in the financial markets, spreadsheets are operating at, close to, or even beyond the present technological limits of their size and/or complexity. There were several reports of the present 256 column limit unnecessarily limiting the number of instruments in a financial portfolio and constraining the level of detail in temporal models. There were reports of difficulties in spreadsheets over 50Mb, a size which is not at all uncommon. Note that spreadsheets >1Gb already exist.

There is very firm evidence from the author’s interviews that people who create or modify spreadsheets in the financial markets are almost entirely self-taught.

4.3 Private Finance Initiative and Public Private Partnerships

The Private Finance Initiative in the UK is the means by which the government now arranges for schools, hospitals and prisons etc to be built, owned and operated. It accounts for approximately 12% of UK government capital spending. Since inception in the mid-nineties approximately 500-600 projects have reached financial close, with many more projects in progress. PFI capital spending is worth several tens of billions of pounds.
Spreadsheets play a central and critical role in this process as they bring together in one place the essential characteristics of the legal agreements for building, staffing, heating, lighting, securing, maintaining and otherwise owning and operating the project in exchange for the single annual charge – the Unitary Charge – that the government pays:

“There is no more basic question in business than how much one should charge for one’s product or service. That question could not be answered in PFI projects without detailed spreadsheet based financial modelling”

“The financial model” is a defined term in the legal agreements relating to PFI projects. Legal agreements specify what the financial model must do, who is responsible for it and how often it should be run or re-run over the 20-30 year project life. There is an issue regarding the ability to continue to run spreadsheets over a 30 year period. There is an aftermarket in the re-implementation of financial models to support the day to day management of the project post financial close.

The requirements specifications for spreadsheet based financial models in the UK’s privatised rail industry were identified as being particularly good:

“The language used in the ITT’s with regard to model specification sets a very high standard”

Bids for rail privatization projects are non-compliant if they fail to meet the specified spreadsheet modelling standards. Within Rail privatisations, there has been a “significant elevation of the status of the model” within the last three years.

The spreadsheet modelling process in PFI (and other forms of limited recourse financing) is now relatively mature, as models and modelling methods have converged to support the generally consistent requirements of the various types of project. Spreadsheet models are extensively checked for correctness using cell-by-cell inspection and other methods [Ettema et al, 2001].

Approximately 1,000 people per year receive specialist training in the City of London in financial modelling skills relevant to PFI and investment banking – 0.3% of “City Type” Employees and 1% of “City Type” employees thought to be working with Important, Key or Critical spreadsheets.

4.4 Law and Maritime

The primary stock in trade of the Law is of course words rather than numbers, and this is reflected in the relatively low-key use of spreadsheet technology within the profession. Spreadsheets are in use where one might expect, in calculating personal injury claims or preparing accounts for example, but they do not play anything other than a peripheral role. A lawyer suggested that spreadsheets were often used to get the right answer ie to calculate the right assumptions to support significant conclusions already reached, a practice which the author has seen from time to time. There were 16,368 solicitors practicing in the City of London in 2003 [Law Society, (2003)].

Use of spreadsheets within Maritime is more widespread as they are used for data storage, data manipulation, the calculation and conversion of charter rates and tonne-mile costs. There is widespread use in the preparation of accounts and financial forecasts. The recent emergence of a shipping derivatives market has seen the very occasional use of
spreadsheets for calculating Value-at-Risk using monte-carlo simulation. Maritime accounts for approximately 10,000 jobs in the City of London, across ship broking, ownership and insurance.

5 SUMMARY AND CONCLUSION

The intensity of spreadsheet use varies across the City of London as one would expect.

Within certain large sectors, spreadsheets play a role of such critical importance that without them, companies and markets would not be able to operate as they do at present.

In only a few of these market sectors are the risks relating to the use of spreadsheets relatively well controlled - in clearing banks, professional services, private finance and investment banking for example.

By contrast, in other sectors where spreadsheets play a Critical role, their use is relatively uncontrolled – the financial markets, fund management, investment research and financial reporting. In these sectors the real risks of material error are not known, and in some cases the risks are vehemently denied. In the worst instances, and supported by interviewee evidence, the existence or use of spreadsheets is concealed by senior management.

In sectors where spreadsheets play merely a Key role – in the internal management of large corporations for instance - spreadsheet use is almost completely uncontrolled. This directly and causally exposes large corporations to significant losses which regularly occur and are publicly reported [EuSpRIG, 2005].

By way of example, during the course of conducting the present 23 interviews, two interviewees each disclosed without prompting (as this was not a research objective) a current and recent instance of where material spreadsheet error directly involving or concerning them had led to adverse events involving “many tens of millions of pounds”.

The situation is reminiscent of the time prior to the unexpected collapse of Long Term Capital Management (LTCM), where there was faith in the infallibility of Nobel Prize winning people and methodologies.

“At the beginning of 1998, the fund had equity of $5 billion and had borrowed over $125 billion — a leverage factor of roughly thirty to one. LTCM’s partners believed, on the basis of their complex computer models, that the long and short positions were highly correlated and so the net risk was small.”[LTCM, 2005]

Spreadsheets have been shown to be fallible, yet they underpin the operation of the financial system. If the uncontrolled use of spreadsheets continues to occur in highly leveraged markets and companies, it is only a matter of time before another “Black Swan” event occurs [Taleb, 2001], causing catastrophic loss. It is completely within the realms of possibility that a single, large, complex but erroneous spreadsheet could directly cause the accidental loss of a corporation or institution, significantly damaging the City of London’s reputation.

As it is impossible to imagine the City of London operating without the widespread use of large spreadsheets, it is important to encourage firms to identify, prioritise, test, correct and control Key and Critical spreadsheets. Doing so will enable firms to improve
profitability by avoiding the inevitable occurrence of large spreadsheet related mistakes, of which only the smallest and least embarrassing are being reported.

Other actions that can be taken include: educating the market regarding the financial, regulatory and other risks of uncontrolled spreadsheet use; improving the regulatory framework regarding their use; promoting the emerging discipline of spreadsheet engineering; training management and staff to build & test more robust spreadsheet models; deploying software tools for detecting and correcting errors; deploying software tools to assist in the management of spreadsheets; encouraging the development of next generation spreadsheet technology and investigating other tools that have the advantages of spreadsheets but fewer or different disadvantages.

6 ACKNOWLEDGEMENTS

The author gratefully acknowledges the generous cooperation of the 23 interviewees without whom this paper would not have been possible. The author thanks a number of interviewees and the anonymous referee for their constructive comments.

7 REFERENCES


CEBR, (2005), “The City’s Importance to the EU economy 2005”, Centre for Economics and Business Research, Unit 1, 4 Bath Street, London, EC1V 9DX.


Reed, C. (2005) Personal communication from Chris Reed, Professor of Electronic Commerce Law, Queen Mary University of London


Regulation and the Integrity of Spreadsheets in the Information Supply Chain

Ralph Baxter,
ClusterSeven, 10 Fashion Street, London E1 6PX
rbaxter@clusterseven.com

ABSTRACT

Spreadsheets provide many of the key links between information systems, closing the gap between business needs and the capability of central systems. Recent regulations have brought these vulnerable parts of information supply chains into focus. The risk they present to the organisation depends on the role that they fulfil, with generic differences between their use as modeling tools and as operational applications.

Four sections of the Sarbanes-Oxley Act (SOX) are particularly relevant to the use of spreadsheets. Compliance with each of these sections is dependent on maintaining the integrity of those spreadsheets acting as operational applications. This can be achieved manually but at high cost. There are a range of commercially available off-the-shelf solutions that can reduce this cost. These may be divided into those that assist in the debugging of logic and more recently the arrival of solutions that monitor the change and user activity taking place in business-critical spreadsheets. ClusterSeven provides one of these monitoring solutions, highlighting areas of operational risk whilst also establishing a database of information to deliver new business intelligence.

1. INTRODUCTION

Section 2 of this paper describes the role that spreadsheets play in business-critical operational environments, whilst Section 3 discusses the difficulty of eliminating them. This leads to an understanding of the generic risks presented by spreadsheets in Section 4 and the implications for the control requirements of SOX (Section 5). Section 6 then addresses the aspects of integrity that must be managed, with Section 7 describing the technical solutions that can assist.

2. SPREADSHEETS IN THE INFORMATION SUPPLY CHAIN

Recent regulation has focused attention on the way that information technology supports end-to-end business processes (for example, PricewaterhouseCoopers, 2004). (In this context an end-to-end business process is seen as a set of business activities that complete the loop from the first initiation of an activity (e.g. a customer placing an order) to the closure of that activity (e.g. cash in the bank).

This commonly demonstrates that the ‘marriage’ between systems and processes is not perfect (see Figure 1). Instead of one end-to-end IT system (or integrated set of systems) supporting the whole process, there are usually gaps where information is retrieved from one or more systems and manipulated before being transferred to the next system in the process.
The management of information in the gaps between formal systems is clearly critical to the integrity output of the information chain – it being only as strong as its weakest link. End user computer applications, particularly spreadsheets tend to be one of the most pervasive support tools at these points (Howard, 2005). It is these spreadsheets – used as operational applications (i.e. supporting regular business tasks) – that are the main regulatory focus.

3. WHY DO SPREADSHEETS PERSIST?
Given the large technology investments made by businesses it begs the question as to why they continue to use spreadsheets to support their information supply chains. The reality is that central systems can only be as good as their original specifications. Since business needs are always changing, the lag between specification and roll out will be at least 6 months (and commonly much more). Hence, in practice, the central systems will always be out of date.

The difference between business needs and central systems capability (Figure 2) is most commonly filled by spreadsheets. The number of these applications reflects the rate of evolution of business needs (more change equals more spreadsheets) and the age of, and investment in, existing systems (older systems and less recent investment leads to more spreadsheets). Indeed, most new financial functionality is templated in spreadsheets before migration to formal systems – which may take years. It is therefore not surprising that much of the competitive edge of businesses (e.g. new financial products) is held in spreadsheet applications.
These aspects of business practice have been implicitly accepted for many years. However, the growth of regulatory concern has brought them to the attention of compliance and audit functions. This can result in policies to eliminate spreadsheets through migration to central systems. Inevitably, however, some are too volatile to be immediately or too complex to be replaced on a cost-effective basis. Other policies have attempted to ban the use of spreadsheets as part of business-critical activity – only to find that the business then does it anyway. The reality is that for many organisations elimination for any significant period of time is nearly impossible to achieve.

Other solutions must be found that can satisfy control requirements for risk management without eliminating the business benefits of spreadsheets. It is therefore necessary to understand the way in which spreadsheets are used in large organizations.

4. SPREADSHEET RISK
In understanding the business use of spreadsheets it is instructive to consider the difference between their use as a modeling tool versus their use as an operational business application. In reality these aspects represent end members of a spectrum but such a differentiation helps to identify the kind of risks that will exist and the most efficient way of mitigating them.

Table 1: Comparison between Spreadsheet Usage as Modeling Tools versus Operational Applications

<table>
<thead>
<tr>
<th></th>
<th>Modeling Spreadsheets</th>
<th>Operational Spreadsheets</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>User</strong></td>
<td>Typically built and used by the same individual</td>
<td>Typically built by developers before being moved across to other individuals or parts of the business for usage.</td>
</tr>
<tr>
<td><strong>Persistence</strong></td>
<td>Models (often extremely complex) may be built over days or weeks, only to be made redundant soon after the relevant business decision has been made.</td>
<td>Models, both simple and complex become part of the critical information flow of the business. They commonly persist for many months or years.</td>
</tr>
<tr>
<td><strong>Structural/Functional Volatility</strong></td>
<td>High. There are likely to be substantial structural revisions from one day to the next as major elements of the model are added or replaced.</td>
<td>Low to medium. All of the key structural elements are likely to be in place. Further evolution of the business process will require maintenance changes – but only rare structural overhauls.</td>
</tr>
<tr>
<td><strong>Data Volatility</strong></td>
<td>Medium. Primarily related to the exploration of alternative scenarios.</td>
<td>High. As the application is relatively mature the transactional data becomes the key variables within the spreadsheet.</td>
</tr>
<tr>
<td><strong>Usage</strong></td>
<td>Likely to be intensive for a short period of time. Usage commonly restricted to a single individual or a small close-knit modeling team.</td>
<td>Usage will depend on the individual business process being supported. Some will be hourly or daily. Others may only feature at week or month end. Usage will involve handover between multiple individuals fulfilling different tasks – not necessarily in the same department.</td>
</tr>
</tbody>
</table>

Table 1 illustrates that the primary risks in spreadsheets used for modeling are related to potential logic flaws in the creation of the workbook. However, in these business
environments the user commonly has a good understanding of the spreadsheet structure and of what answers ‘make sense’. Since decisions are based on multiple iterative scenarios the risks of decisions based on incorrect processing are relatively low. For these reasons modeling spreadsheets receive less attention from regulators (Buckner, 2004).

In contrast, the risks attached to operational spreadsheets depend on the ongoing maintenance of the logic integrity in the spreadsheet. These risks are increased by such factors as multiple users and their lack of detailed knowledge of the spreadsheet structure. Risks are also increased where the output is the aggregation of many transactions where it is unlikely that anybody has a good understanding of what the ‘right number’ ought to be, even though it may be a key input to financial control processes. It is for these reasons that operational spreadsheets are much more the focus of regulatory concern.

5. REGULATORY REQUIREMENTS
Sarbanes-Oxley represents just one face of the most recent focus on the operational risk and financial reporting of corporations. Whilst the specific requirements of each piece of regulation may be open to interpretation, the overall direction is clear: to ensure that businesses understand what is happening in their organisation; to be able to respond in a timely fashion to issues when they emerge; to have procedures in place that minimize the possibility of things going wrong in the first place; and, most personally, to hold the business and its key executives to account if they don’t do it.

Executives will have to gain a deeper understanding of internal controls because their business decisions will be placing greater reliance on adequate internal controls and ensuring that they are deployed, maintained, adjusted and reported on as required. John Flaherty, former COSO chairman, said that this means “… that every division in a company needs to have a documented set of internal rules that control how data is generated, manipulated, recorded and reported …”

For SOX there are four sections most relevant to spreadsheets and their controls:

Table 2: SOX Implications for Operational Spreadsheets

<table>
<thead>
<tr>
<th>Section</th>
<th>Requirements</th>
<th>Spreadsheet Implications</th>
</tr>
</thead>
<tbody>
<tr>
<td>103 Auditing, quality control, and independence standards</td>
<td>Independent auditors must include an evaluation of the Company’s internal controls in their report. The evaluation will include a description of material weaknesses in internal controls and material non-compliance with them.</td>
<td>Un-monitored spreadsheets in the critical information supply chains will fail this test.</td>
</tr>
<tr>
<td>302 Corporate responsibility for financial reports</td>
<td>Executives must evaluate the effectiveness of internal controls every quarter. Financial reports must include their conclusions about internal controls and explain any significant changes to them. All frauds, no matter how small, must be disclosed to the Company’s auditors and to the Audit Committee of the Company’s Board of Directors.</td>
<td>It may be possible to eliminate spreadsheets temporarily but unlikely they can be eliminated for every quarterly report. Spreadsheets are also a common source of fraud.</td>
</tr>
</tbody>
</table>
It is clear from the risks identified above that spreadsheets have potentially significant failings against regulatory demands and more general tests of business control:

- They are highly vulnerable to error and, occasionally, fraud
- The information they contain and the user interaction with them are not transparent to the rest of the organisation.
- It takes significant time and effort to understand unexpected changes and to respond and communicate them as appropriate

In order to resolve these challenges organizations must use processes and technology that can ensure the integrity of business critical spreadsheets. Solutions may vary from entirely manual to strongly technologically enabled but all must focus on the possible causes of losing integrity of the spreadsheet output.

### 6. WHAT GOVERNS INTEGRITY?

Full spreadsheet integrity (i.e. assurance that the output is the expected processing of the input) is dependent on five key elements:

1. That the programmer of the spreadsheet logic model correctly understands the transactional process to be implemented (i.e. correct specification)
2. That the programmer has created the required logic without errors (i.e. no bugs)
3. That subsequent data inputs are valid (whether manual or automatic)
4. That subsequent user and maintenance activity does not corrupt the original logic.
5. That where multiple user tasks are performed on a spreadsheet these are performed in the correct order.

Although these challenges are relatively short to define the pervasiveness of spreadsheets and their almost infinite flexibility means that solutions have taken much longer to emerge.
It is also apparent that the ‘culture’ around spreadsheet within many large organizations does not contribute to avoiding or removing these problems. For example spreadsheets are commonly seen as a temporary solution that will be replaced at the appropriate time by ‘proper’ investment in fully architected systems – and by implication it is not sensible to make this investment in the spreadsheet application itself. This applies even in organizations where some spreadsheet models have persisted for many years. It is ironic that spreadsheets are often viewed as a tactical solution when they are one of the longest standing parts of most enterprise information systems.

7. HOW CAN TECHNOLOGY HELP?

Several tools have been created to address the potential flaws in logic creation. These include Spreadsheet Professional, HMC&E SpACE, Operis OAK, recent Microsoft Excel 2003 error tips and others. Clearly the wide variety of spreadsheet structures and logic requirements means that none of these tools can be 100% effective. They are reliant on looking for inconsistencies in successive cell formulae or performing checks for logic elements that are known to be particularly error prone (e.g. nested IF formulae).

Despite the research demonstrating the prevalence of errors in almost all spreadsheet logic it is clear that with (and without) logic-checking tools most organizations feel comfortable that they can get their operational spreadsheets to be reliable at particular points of time (e.g. testing and audit). The next question is how to maintain this quality after the spreadsheet enters operational usage.

Perhaps the most common solution is to impose some form of lockdown on spreadsheet change. This can be effective in highly resourced environments (where developers are on hand to change, test and re-issue a revised version within the timetable of business needs) or where the spreadsheet application has become very mature in its usage (i.e. business needs are not changing). However, in less resource-rich or mature environments this policy inevitably fails because it prevents the user exercising their own business knowledge to resolve their own problems. As a result it is almost always circumvented.

A second option is to continue using the logic tools in the operational environment. However, these tools are usually inappropriate for such use as they are designed to look for logic inconsistencies rather than track broader categories of data behaviour. Moreover, they would have to be utilized after every user interaction (probably impractical) and must be interpreted by someone who understands the underlying structure of the spreadsheet (less common in operational spreadsheet usage).

A third option is now appearing. These are solutions designed to specifically address operational usage of spreadsheets. ClusterSeven is one example of these new solutions that focuses on the change management and user interaction with business critical spreadsheets. In so doing ClusterSeven can expose all of the time variant information contained within spreadsheets – be it data, functionality or usage. This allows it to highlight areas of operational risk (when activity does not conform to expected patterns) and also to create business intelligence (such as reporting on the trends of data values for particular key performance indicators).

Given the low take up of existing technology tools (compared to the pervasiveness of Excel) it is appropriate to ask whether yet more technology will provide the answer. ClusterSeven believes that a number of factors are now converging to make this a reality: firstly auditors
and regulators are becoming increasingly vocal that the status quo is not satisfactory; secondly the combination of logic tools and operational tools enables the whole spreadsheet lifecycle to be managed and thirdly there is growing acceptance (all be it grudging in places) that spreadsheets are here to stay, necessitating a strategic approach to the problem.

8. CONCLUSIONS

- Spreadsheets populate the information supply chains of many large organizations.
- As the ‘weakest link’ in the information supply chain they have become the target of regulatory concerns about financial reporting.
- Regulatory concerns can be addressed by adopting processes that ensure the continuing integrity of key operational spreadsheets.
- Integrity can be maintained through the application of arduous manual processes or via the assistance of technology.
- Integrity also requires a shift in the culture of organizations to see spreadsheet technology as a persistent strategic part of their infrastructure rather than a short term tactical fix.

9. REFERENCES


Howard, P (2005), Spreadsheet Management, A Briefing Paper by Bloor Research

PricewaterhouseCoopers (2004), The Use of Spreadsheets: Considerations for Section 404 of the Sarbanes-Oxley Act.
ABSTRACT

Qtier-Rapor is a framework that enables a business to control, automate and manage spreadsheet processes. This paper is a discussion document on the key features within Qtier-Rapor that correspond to the requirements of a Sarbanes-Oxley audit, in order to demonstrate that compliance and the use of Spreadsheets are not mutually exclusive.

There is a widely held view that business information systems are secure, but that once data is extracted into spreadsheets to consolidate a corporate view, then most of the security evaporates and, if that data is business critical, then compliance evaporates with it.

Whilst no single product can be a complete compliance solution, Qtier-Rapor provides a methodology that, when combined with appropriate procedures, enables Excel spreadsheets to be integrated with corporate data with all of the security, integrity and auditability necessary for compliance with the Sarbanes-Oxley Act.

1 INTRODUCTION

The Qtier-Rapor framework oversees the security and flow of data through integrated spreadsheet systems. Spreadsheet templates are held in a secure repository, and spreadsheet derived data, can be held in secure, structured, databases to which access can be controlled using formal IT methods. Processes to control data access, input, validation, consolidation, and reporting are all designed in simple steps.

The control of spreadsheet templates and data, along with control of Excel functions such as ‘Save’ & ‘Print’, enable spreadsheet systems to be designed for compliance, and to remain compliant during use.

This document discusses the key features of Qtier-Rapor and the way in which they relate to the requirements of a Sarbanes-Oxley 404 audit.

2 IN A SARBBOX ENVIRONMENT, IS IT MANAGEMENT VS USERS ?

Section 404 of the Sarbanes-Oxley Act requires management of SEC registered companies to report on the effectiveness of internal controls over financial reporting.

Section 404 also requires the company’s independent Auditors to attest to and report on management’s assessment of the effectiveness of these internal controls.
"Sarbanes-Oxley implies managers can't ignore un-controlled spreadsheets" [Pettifor B, Eusprig Conference 2003]

"The presence of a spreadsheet application in an accounting system can subvert all the controls in all other parts of that system" [Butler R. 2000]

'End users are putting their companies at risk by setting up spreadsheets without realising that this demands the discipline of traditional programming. Our [KPMG] 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' [Kavanagh J. 1997]

"In order to comply with the Sarbanes-Oxley Act we can no longer use spreadsheets within our business" [Many CFO's and CIO's. 30 July 2002 onwards]

"You can take my spreadsheets from me when you prise them from the fingers of my cold, dead, hands!" [Most Excel users. 2005]

3 SARBANES-OXLEY AUDIT CONSIDERATIONS

A Sarbanes-Oxley Audit Plan should deal specifically with the auditing and control of spreadsheets. The PriceWaterhouseCoopers document 'The Use of Spreadsheets: Considerations for section 404 of the Sarbanes Oxley Act' illustrates the considerations that auditors must give to spreadsheets.

It says that spreadsheets may be very complex, relying heavily on macros, input from linked supporting spreadsheets, and complicated calculations. In these instances such spreadsheets should be treated as software applications in their own right.

The Audit Plan should address the following key areas:

**Change control** – maintaining a controlled process for requesting changes to a spreadsheet, making changes and then testing the spreadsheet and obtaining formal sign off from an independent individual that the change is functioning as intended.

**Version control** – ensuring only current and approved versions of spreadsheets are being used, by creating naming conventions and directory structures.

**Access control** – limiting access at the file level to spreadsheets on a central server and assigning appropriate rights. Spreadsheets can also be password protected to restrict access.

**Input data** – ensuring that reconciliations occur to make sure that data is inputted completely and accurately. Data may be inputted into spreadsheets manually or systematically through downloads.

**Security and integrity of data** – implementing a process to ensure that data embedded in spreadsheets is current and secure. This can be done by locking or protecting cells to prevent inadvertent or intentional changes to standing data. In addition, the spreadsheets themselves should be stored in protected directories.

**Documentation** – ensuring that the appropriate level of spreadsheet documentation is maintained and kept up to date to understand the business objective and specific functions of the spreadsheet.

**Development lifecycle** – applying a standard Software Development Life Cycle to the development process of the more critical and complex spreadsheets covering standard phases: requirements specification, design, building, testing and maintenance. Testing is
a critical control to ensure that the spreadsheet is producing accurate and complete results.

**Back-ups** – implementing a process to back up spreadsheets on a regular basis so that complete and accurate information is available for financial reporting.

**Archiving** – maintaining historical files no longer available for update in a segregated drive and locking them as “read only”.

**Logic inspection** – inspecting the logic in critical spreadsheets by someone other than the user or developer of the spreadsheet. This review should be formally documented.

**Segregation of duties / Roles and procedures** – defining and implementing roles, authorities, responsibilities and procedures for issues such as ownership, sign off, segregation of duties and usage.

It is immediately evident that, even if such controls are imposed, normal spreadsheets can be made to fail in almost every area during use; particularly by a user with malicious intent. It should be noted that Sarbanes-Oxley was enacted in the wake of a number of financial reporting scandals where key business data was in error or was manipulated at the very top level of the Corporation.

### 3.1 Change Control

Change controls are circumvented if a spreadsheet can just be amended and resaved under the old name. Within the Quiq-Rapor framework, spreadsheet templates are held in a secure repository. Only authorised ‘administrators’ can access the template to modify the design or embedded logic.

Access is security logged automatically, and the new version is quarantined as “suspended” until tested, audited and authorised for release.
3.2 Version Control

Many versions of a spreadsheet, all with the same name, can exist within a business. As can be seen in Fig 1, old versions are withdrawn and version numbers are incremented when processes are modified. Only the latest approved version of a process is available to users, and previous versions are archived so that the functional logic in use at any time in the past can be examined.

3.3 Access Control

It is quite possible for a person with malicious intent to find a spreadsheet on a server, and software is readily available across the internet to decrypt spreadsheet protection passwords.

Qtier-Rapor controls access through menu structures (Fig. 2), and ensures that only those spreadsheets to which the user has authorisation will appear on the menu. If a user finds a spreadsheet template on the server and tries to open it, the template 'knows' that it is controlled by the Qtier-Rapor framework (Fig. 3) and still validates access against the user's logon. The spreadsheet is also password protected by a varying, system allocated, password.

Fig. 2

Even if the password is decrypted in one instance, and the logic changed, it cannot then overwrite the original template repository without authority.
3.4 Input Data

Any system is only as good as the data that is input. Qtier-Rapor ensures that any data that can be obtained automatically from corporate data files is made available to any spreadsheet process without user intervention. This ensures that all users access the same data and that there is ‘only one version of the truth’.

Data derived from a process can be written to Qtier-Rapor’s User Defined Database (UDD) for retrieval by subsequent processes. Dependency checking ensures that processes reliant on the completion of previous operations (such as consolidation) cannot progress until those operations are completed.

All spreadsheets can contain validation ranges on input cells, but sometimes entries can be individually valid, but give rise to a combination of circumstances that are invalid. Qtier-Rapor can maintain global ‘reasonable ranges’ of expected results against key indicators within the UDD. Spreadsheet processes can then perform a reasonableness validation against those key indicators and take appropriate action. These global validation checks can be absolute, or allow values to fall within a range, with appropriate logic embedded in the step design to respond to out-of-range results (Fig. 4).

This test of reasonableness has been previously identified by EuSpRIG as a key factor in validation of data entry in spreadsheets.
3.5 Security and integrity of data

Only one current spreadsheet template can exist at any one time for any one process, and it is secured against tampering by locking with a system generated password. This ensures that the process logic is unchanged since it was audited and released.
Each template can access data direct from the corporate database, or from the User Defined Database (UDD), using formal IT methods through a Connections and Catalogues system. This ensures that the data is absolutely current. The spreadsheet templates are themselves stored in a secure repository.

All actions are audit logged showing the process used, criteria selected at the time, user, computer ID and timestamp. (Fig. 5)

3.6 Authentication
Rigorous authentication of printed output from spreadsheets is not normally possible. It is easy to save a copy of a locked spreadsheet, or even copy and paste it onto a new sheet, then amend the formulae, change the data and re-type any timestamp information to make it look exactly like the original version.

Qtier-Rapor can disable the Save, SaveAs, and Cut&Paste options in Excel. A special feature can embed a unique, non-reproducible, authentication stamp into any printed material, that corresponds to the security log entry.

This means that spreadsheets cannot be duplicated (other than by complete re-typing), and any screen-prints or cut & pasted ‘copies’ cannot have the authentication stamp. The management, and auditor, can be assured that the report being examined is a validated output of the system.

3.7 Documentation & Development Lifecycle
Qtier-Rapor is self documenting, containing the detail of each step and logic diagrams for each process. This does not, of course, include documentation of logic embedded in the spreadsheet. Process documentation is updated along with the spreadsheet design. The framework lends itself to the normal IT disciplines of application development.

3.8 Backups & Archiving
Creating back-ups and archiving old spreadsheets places a heavy reliance on user initiated processes. Critical data sits in a number of spreadsheets, in a variety of locations. It is rarely possible, without extensive effort, to ensure effective back-ups.

The Qtier-Rapor framework saves data into structured IT databases in known locations. These can be backed up and archived on the server, as in any IT application, without relying on the user.

3.9 Logic Inspection
Qtier-Rapor instils the discipline of best practice programming into 'power Excel users'. This is achieved by dividing processes into small steps and assembling them within a visually displayed logic diagram. Peer review is made easier, as is audit.

No claim is made that Qtier-Rapor can validate spreadsheet logic – that is the province of other tools and techniques available. However, once logic processes are embedded they are locked and not subject to amendment unless under change and version control.

3.10 Segregation of duties / Roles and procedures

Normally, spreadsheets do not know or care who uses them. The corporation can design an authority matrix, but that will not ensure that the procedures are followed. With Qtier-Rapor, every spreadsheet process knows and cares who is (trying to) run it.

Fig. 6

Roles are maintained which give authority to processes (Fig. 6) and to which Users can be allocated. This extends not only to use of processes, but also to the security, design & release functions. All of the above is subordinate to security policies implemented on the corporate servers.

Consequently, a defined procedure can be implemented that clearly states, and controls, the authority of individual users, ensuring (for example) that a designer cannot be responsible for sign off and release of a process.
4 SUMMARY

Excel spreadsheets are uniquely flexible to meet the demands of modern accounting and reporting, where information is needed quickly, and adaptability is the key to business speed.

However, uncontrolled flexibility and the ability to manipulate data manually, means that no information has provenance – But provenance is what is demanded by legislation that seeks to instil rigid process and control over financial reporting. One of the biggest holes in 404 compliance occurs when business-critical information is held in spreadsheets.

Many corporations that need to demonstrate compliance to the Sarbanes-Oxley Act are giving consideration to banning the use of spreadsheets for critical data areas. They are implementing reporting tools with a rigid pre-defined structure – but where does the data originate? How many companies don’t feed their budgets from spreadsheets? How many don’t extract their forecasts to spreadsheets?

How to square the circle? Users need to retain flexibility by retaining Excel as their working tool – But management needs to control, authenticate and monitor the logic, inputs and outputs to the same degree as a formal IT system.

The Qtier-Rapor framework wraps itself around the whole spreadsheet process to provide all of the elements necessary to allow users the flexibility of a spreadsheet interface, but to deny them the opportunity of breaking the provenance and integrity of the business-critical data that they produce.

Sarbanes-Oxley compliance is about making executives more accountable for their information.

Sarbanes-Oxley IT audits are about ensuring systems are more secure to support compliance.

Qtier-Rapor is about making Excel spreadsheets as compliant as any other IT system.

5 FUTURE PERSPECTIVE

Use of spreadsheets as a tool is a ‘must have’ for anyone involved in finance. Imposition of additional workload on any aspect of the use of such a tool will encourage users to look for ways to subvert the process. Qtier will continue to enhance the useability and power of functions, in order to ensure that design of spreadsheet systems is undertaken within the Qtier-Rapor framework as a matter of choice rather than mandate.

Current development includes:
- Dissemination of spreadsheet reporting & ad-hoc data access through a web browser.
- Version control & access control for existing (non-Rapor) spreadsheets.
- More extensive anti-tampering techniques for spreadsheets distributed outside of the framework by email.
6 REFERENCES

'The Use of Spreadsheets: Considerations for section 404 of the Sarbanes-Oxley Act'. Available for
download from PriceWaterhouseCoopers 'Sarbanes-Oxley Information Centre at http://www.cfbdirect.com,
14:08 8/03/2005.


Kavanagh J., article in Computer Weekly, July 1997

ABSTRACT

Contemporary spreadsheets are plagued by a profusion of errors, auditing difficulties, lack of uniform development methodologies, and barriers to easy comprehension of the underlying business models they represent. This paper presents a case that most of these difficulties stem from the fact that the standard spreadsheet user-interaction paradigm – the 'cell-matrix' approach – is appropriate for spreadsheet data presentation but has significant drawbacks with respect to spreadsheet creation, maintenance and comprehension when workbooks pass a minimal threshold of complexity. An alternative paradigm for the automated generation of spreadsheets directly from plain-language business model descriptions is presented along with its potential benefits. Sunsight Modeller™, a working software system implementing the suggested paradigm, is briefly described.

1 INTRODUCTION

Even the most cursory perusal of journal literature on electronic spreadsheets in recent years will reveal that its dominant theme is consistently plaintive. The litany of dissatisfaction with the current state-of-affairs in the general use of spreadsheets includes the following:

1) Profusion of spreadsheet errors. There are a vast collection of studies on the prevalence of errors in spreadsheets [Panko, 1996]. Many studies [Chadwick, 2000; KPMG, 1997] consistently report that upwards of 90% of spreadsheets that pass a certain threshold of complexity may contain major errors (defined as errors that could affect decisions based on the results of the model), most of them due to individually seemingly trivial actions such as erroneous cut-and-paste, incorrect cell-references, obsolete absolute cell-references and structurally disruptive column-and-row alterations. EuSpRIG maintains an entire Web-page devoted to tracking press reports of spreadsheet errors. Given that the preponderant use of spreadsheets in professional activity is related to business and financial reporting and decision-making, the total cost of spreadsheet errors is enormous.

2) Auditing difficulties. A study by three PricewaterhouseCoopers researchers [Ettema et al, 2001] reports on the cost of auditing a typical spreadsheet in the following manner: The number of distinct formulae depends on the nature of the spreadsheet, but for moderately sized spreadsheets this number typically lies in the range of 500 to 1,500. The inspection of a distinct formula by an experienced auditor takes on average around 3 minutes. The total effort of a traditional spreadsheet audit generally
takes from 25 up to 75 hours. If the number of distinct formulae is much larger
1,500, or if the spreadsheet is complicated and badly structured, it can be difficult to
perform an audit by more than one auditor in parallel. The throughput time for
auditing such spreadsheets will be, based on these assumptions, more than a
fortnight. Given these numbers and typical corporate time pressures, spreadsheet
audits are too often performed in a cursory manner, if at all. The attendant risks,
especially in light of the requirements of the Sarbanes-Oxley act, are alarming

3) **Absence of Documentation**. The absence of documentation has been a factor in a
number of well-documented spreadsheet errors [Butler, 2000]. Failure to document
can lead to serious errors and maintenance nightmares, especially in environments
where a model is passed from one user to another.

4) **Lack of Uniform Development Methodologies**. It appears that most organisations
do not have even the most rudimentary internal modelling standards [Chadwick,
2000]. The lack of uniform corporate-wide standards, methodologies and
presentation styles hinders quality control and multiplies confusion.

5) **Re-use is Nearly Non-existent**. [Ettema et al, 2001] report that PWC has collected a
wide collection of formulae that appear over and over again in many spreadsheets.
Efficiency interests would indicate that these formulae ought to be re-used from
already existing spreadsheets by spreadsheet creators, rather than re-composed from
scratch and re-audited in each new spreadsheet. This is not done because re-use
typically involves cut-and-paste which is tedious and error-prone.

This paper presents the thesis that all of the above problems are related to aspects of the main
spreadsheet interface paradigm – which may be termed the ‘cell-matrix’ paradigm – invented
in the 1970’s and propagated nearly unchanged to this day. The cell-matrix paradigm is
excellent for the presentation of data in spreadsheets, but unwieldy and error-prone with
respect to other actions such as spreadsheet creation, maintenance, model comprehension,
and auditing. It follows that the corrective should involve the introduction of different
interface paradigms. The paper ends with a brief presentation of Sunsight Modeller™, a
commercially-oriented software package that implements in practice the ideas brought forth
here.

2 **AFFORDANCE AND PARADIGMS**

2.1 Affordance

*Affordance* is a word originally coined in the psychological research of James Gibson, but its
most widespread introduction came from books on the design of objects by Donald Norman
[Norman, 1988]. The meaning of the word is perhaps best understood by example. A person
approaches a door designed to appear as a monolithic slab. In which direction does it open?
Should one pull or push, on the left or on the right? Perhaps it slides open? If, on the other
hand, the door has on it a prominent flat handle with an illustration of a human palm, one
would have no hesitation walking up to the handle, placing a hand against it, and pushing the
deroor open. The handle ‘affords’ being pushed.
Based on such examples, a literature on affordance and its application to the design of everything from staplers to pavement paths has emerged. The basic principles of affordance, as stated in the ample literature [Gibson, 1977; Gibson 1979; Norman, 1988; Norman, 1990] can be summarised in a handful of sentences: Take advantage of analogies and cultural standards. Place a stress on human intuition, natural thought-processes, retention, prior experience, natural language, and consistency. Strive to make the most important features salient and obvious.

In the computing context, the idea of affordance has been most widely adopted by professionals working on graphic user interface design, and is often applied with respect to questions such as how a clickable button should appear and where, or in what context to use radio-buttons as opposed to tabs. It can, however, be considered in a more systemic context, as applying to an entire paradigm of interface. The most prominent example of a shift to what may be termed an affordance paradigm is the adoption of the desktop metaphor supplanting the older command-line paradigm. The history of the evolution of computer languages, from machine code to assembler to functional programming to object-oriented languages, may similarly be viewed as shifts of interface paradigm intended to boost affordance by ever more closely mimicking human intuitions and languages, and making central features salient.

2.2 The Cell-Matrix Paradigm

The commercial electronic spreadsheet was invented in the late 1970’s by Daniel Bricklin and Robert Frankston. It represented a seminal event in the adoption of personal computers in businesses and homes, and its impact on modern business methods cannot be overstated. As the name suggests, the electronic spreadsheet took its metaphor from the realm of accounting, where a "spread sheet" meant a large sheet of paper with columns and rows that organised data about transactions for a business person to examine. Bricklin has also been quoted as saying he was influenced by blackboard presentations of data in business school lectures (cf. www.bricklin.com). These conceptual roots form the bedrock of the main spreadsheet interface paradigm, which may be termed the ‘cell-matrix paradigm’. In its very familiar essence, particular cells within a matrix are designated to present data. Data in one cell is inter-related to data in another cell, thus ensuring that the entire matrix presents a conceptual whole, by attaching a formula to a cell, with that formula referring to other cells by code, typically by a combination of row and column designators such as C24 or H86, for example.

From the perspective solely of the dimension of data presentation, the cell-matrix paradigm is an excellent example of applying the principles of affordance in the field of computer automation, which is doubtless a major factor in its phenomenal success. By borrowing its presentation style from the spread-sheets used for centuries in businesses, it took full advantage of analogies, prior experience and cultural standards. Reading a spreadsheet for the purpose of seeing the numbers and taking in ‘the bottom line’ is intuitive for humans, and, albeit with some effort, spreadsheets can be styled to make their most important data appear to be the most salient features. The automation of calculation borne by the cell formulae frees humans from performing tedious and repetitious tasks.

The problems begin because the very same cell-matrix paradigm is also used for the purposes of spreadsheet creation, comprehension and auditing. It utilises the ‘What You See is What You Get’ interface, conflating structure, presentation and content. This type of interface is
quite agreeable in many other situations but fails in the context of spreadsheets. It is a
tolerable and perhaps even an inviting metaphor when one is creating a small spreadsheet, in
a ‘tinkering’ manner. But when spreadsheets grow beyond a certain threshold of complexity,
the strain of retaining in the mind scattered cryptic cell-references in hundreds or thousands
of formulae becomes too much to bear. One may reference in this regard studies on the
importance of distinctions between presentation, data and the underlying logic in the
authoring of computer-related presentations [Isakowitz et al, 1995].

Human beings do not naturally think of business models in terms of cells in a matrix. ‘Net
Income’ bears far more meaning than cell-reference C38. When schools of business
administration and accountancy present business models, they do so using conceptually
related collections of formulae in verbal human language. The mental effort required for
translating between this model and the cell-matrix metaphor may be regarded as the root of
many difficulties. Cell-references that are meaningless in and of themselves are an open
invitation to errors. Filling in every cell in an immense matrix is a tedious, repetitious and
time-consuming task, with copy-and-paste actions serving as a further pitfall for errors.
Reconfiguring a matrix structure to fit an evolving mental model can similarly be tedious and
error-prone. The complexity of large and multiple matrix workbooks coupled with the
wearisome cell-chasing required for elucidating the meaning of formulae is an impediment to
comprehension and drives up the costs of auditing. The traditional spreadsheet does not
afford ample documentation. Reusability of formulae from one spreadsheet to another is
rendered nearly impossible by the strictures of the cell-reference method. The design of the
interface, which involves manual specification of structure, presentation and content, works
against uniformity of presentation and hinders the adoption of corporate-wide methodologies.

Researchers studying the most effective methods for increasing spreadsheet accuracy [Kruck,
Sheetz, 2001] generally recommend careful planning and design of spreadsheets,
simplification of formula complexity and testing of spreadsheets. There is also ample
empirical evidence that many spreadsheets are created in a collaborative manner rather than
isolated individuals [Nardi, 1993]. All of these arguably are hindered by the above-listed
drawbacks of the cell-matrix interface.

2.3 Out of the Shadows

The most obvious partial remedy to the above-presented situation is to use named references
for cells. [Napier et al, 1989] compared the performance of novices using Lotus HAL with
Lotus 1-2-3, and concluded that HAL users consistently solved more problems because the
language more readily allowed reference to spread-sheet cells by names. Named references
for cells have long been available in most commercial spreadsheet software products, and the
intuition that consistent use of named references contributes significantly to reducing errors is
also supported by the study [Janvarin and Whittle, 2003]. It appears, however, that most users
of spreadsheets do not make use of this feature.

In the paper *Spreadsheet assurance by “control around” is a viable alternative to the
traditional approach* [Ettema et al, 2001], a case is presented for utilising what is termed a
‘shadow model’ as an aid for auditing. The shadow model consists of formulae written in
plain English which model the spreadsheet that is to be audited. The audit methodology then
consists of importing scenarios of input data from the spreadsheet and comparing the shadow model’s computed results with the spreadsheet’s output.

In the terms defined in this paper, the shadow model may be interpreted as an interface paradigm that differs from the cell-matrix paradigm and is tailored to serve as a better affordance for certain spreadsheet tasks. Ettema et al note several beneficial effects of working with the shadow model. A partial list of those benefits includes: clean separation of data and calculations; plain language variable names instead of alphanumeric cell-references; and clear access to the logic underlying a spreadsheet that is often otherwise difficult to discern. They also point out that incremental development of models is much easier in the shadow model than in a spreadsheet, because adjustments and supplements are often difficult or risky to incorporate in spreadsheets.

In fact, Ettema et al find these benefits so compelling that they ask ‘Should spreadsheets be used at all’ in a very prominent place in their paper. It would seem that the proper response to that question should be ‘yes’, if only because spreadsheets are extremely well suited for their original, pre-electronic, purpose: immediate presentation of data in tables. But this leads to another question: why should one be content for the shadow model to remain in the shadows? Why not keep the spreadsheet for data-presentation, but adopt the modelling language as an affordance paradigm for spreadsheet modelling, creation and comprehension?

Model Master, developed by Jocelyn Paine [Paine, 2001], takes a step in that direction. Model Master uses a text-based language for programming spreadsheets, which the Model Master compiler converts into actual spreadsheets. The syntax of Model Master, however, resembles that of object-oriented programming languages, which it intentionally mimics. It cannot be expected that the vast majority of business users of spreadsheets will feel at ease working with what to them will look like computer coding. Atebion [Atebion, 2005], another tool utilising an innovative interface, uses a mix of visual programming and natural English. [Nardi and Miller, 1990] note in a study they conducted of spreadsheet users that “the key to understanding non-programmers’ interaction with computers is to recognise that non-programmers are not simply under-skilled programmers who need assistance learning the complexities of programming. Rather, they are not programmers at all. They are business professionals … whose jobs involve computational tasks”.

3 DESIRED FEATURES OF A NEW PARADIGM

Following upon the above, we may now form a list of desired features for a new affordance paradigm for the various actions related to spreadsheets:

1. *A modelling language for the purposes of describing, comprehending and auditing the model expressed by a spreadsheet, whilst keeping the traditional spreadsheet itself for data-presentation.*

1.1. The modelling language ought to be close to natural human language, whilst striving to avoid being so broad that it would permit confusing ambiguity.

1.2. Ideally it should be indistinguishable from the language people use to describe business models when writing on white-boards or in text-books. It should be possible for a person with a reasonably general business education to
comprehend a written model directly, and to learn how to compose a written model rapidly. This would allow a design document to serve double duty as a programming tool and human readable documentation of the model or goal of the spreadsheet.

1.3. It should be tolerant of variations in self-expression when those variations are non-ambiguous.

1.4. The modelling language should be platform-independent – it should be possible to use the modelling language to generate spreadsheets in Excel, or StarOffice, or any other spreadsheet format.

2. A generator that translates ‘time-series models’ – meaning models composed of variables and formulae spread over several time-periods, which constitute the bulk of business models – into spreadsheets.

2.1. The generator ought to free users as much as possible from repetitious and tedious tasks.

2.2. The generated spreadsheet should not contain column-and-row cell-references. The generator should be clever enough automatically to implement named references.

2.3. Being software, the generator, in going from model to spreadsheet, can provide much value-added over hand-crafted spreadsheets – for example, automatically separating data-only variables from calculated variables, presenting trees of dependency relationships between variables, analysing sensitivity rankings, and so forth.

3. Separation of the conceptual model from the presentation structure of the spreadsheets.

3.1. It should be possible to define look-and-feel and general presentation structures separately from the model language.

3.2. This can enable users to create spreadsheets that are uniform in look-and-feel and presentation structure simply by defining these parameters once, freeing them to concentrate on the important details of their models without the distractions of look-and-feel details.

The benefits that can be attained from software that can achieve these broad aims should be clear:

1) **Reduction of errors.** The best preventative of errors is a structural arrangement that reduces their likelihood of occurrence. The elimination of cut-and-paste actions from cell to cell, obscure cell-references and structural column-and-row alterations in the modelling of spreadsheets can be expected to translate into a reduction of entire classes of now common errors. Expressing models in plain language will also facilitate comprehension of formulae on the part of spreadsheet composers.

2) **Ease of auditing.** Reading a human-language description of the model underlying the spreadsheet affords immediate comprehension and can guide auditors to spot assumptions and formulae requiring special attention. The automatic insertion of named-references in cell formulae affords easier cell-by-cell inspection than alphanumeric references.
3) **Ease of documentation and maintenance.** Spreadsheets do not afford documentation. Text files do. A verbal spreadsheet model passed from one user to another, with documentation, is easier to maintain than a manually-created spreadsheet.

4) **Corporate-wide uniformity and re-use.** Utilising this interface paradigm can contribute to attaining the so-far mostly elusive goal of corporate-wide uniformity in spreadsheet lay-out styles and modelling methodologies. From the perspective of spreadsheet presentation, corporations can adopt uniform standards regarding matters such as left-to-right and top-to-bottom ordering of formulae, colour schemes, the location of variables requiring data-entry versus calculated data, and so forth, leaving only the modelling details to be changed as needed by individual projects and employees. With models separated from lay-out, it may be easier to institute corporate-wide instruction and uniformity in modelling methodology, and inculcate modelling cultures that focus on the most important elements of constructing financial models. Corporations can also develop and maintain libraries of formulae that can readily be re-used in models.

5) **Reduction of Sarbanes-Oxley Risk.** Given the provisions of the Sarbanes-Oxley Act, officers of corporations who cannot attest to the fact that they have reviewed financial statements, assured that they contain no untrue or misleading statements, and instituted internal controls to ensure the integrity of corporate financial statements may be liable for serious penalties. A modelling-language centred approach to generating spreadsheets can do much to mitigate this risk, in several ways. Simply having the ability to read a plain-language description of the model underlying a spreadsheet grants executives the ability to attest they have clear insight into the financial assumptions underlying corporate activities. Because the generated spreadsheets are separate from the verbal model, they can be locked from unauthorised editing, with cells calculated by formulae locked against over-writing.

### 4 AN IMPLEMENTATION: SUNSIGHT MODELLER™

Sunsight Modeller™ 1.0, produced by Inrise Financials, Inc., is a software system that implements the desired affordance paradigm for business-oriented spreadsheet creation, auditing and maintenance that has been put forward in this paper. It is intended for multiple uses, including scenario modelling, optimisation, research and calculation. We present here a very brief description of the main points of Sunsight Modeller™ 1.0 and some of its features.

#### 4.1 Business Algebra Modelling Language

In order to use Sunsight Modeller™ 1.0 to create a spreadsheet, a user writes out what may be termed a ‘Business Algebra Model’, which is essentially a plain-language description of what the generated spreadsheet ought to contain. The Business Algebra Model may be composed in Microsoft Word or indeed virtually any text editor. The user would normally go through the following steps:

1. **Define the ‘time-frame’ which will appear in the spreadsheet.** This requires expressing, in words, the time period length, the number of periods and when the first period begins. A time frame might therefore look like:
Each period is one year.
The number of periods is 7.
The first period starts in 2005.

2. Optionally define ‘outline categories’. Business models frequently require dividing variables amongst several categories. For example, sales numbers may be gathered separately based on products and markets, and then summed for presenting general totals. Sunsight Modeller™ 1.0 enables one to generate spreadsheets with these sort of category breakdowns very easily, and automatically sets outlines and calculates summed ‘roll-ups’ over the categories. The category hierarchies need be written only once, and after that only the title of each hierarchy has to be referenced in reports in order to have every variable broken down into the elements of the categories.

This done by writing out ‘Categories:’ and then writing an outline representing the hierarchies of categories, with the top or ‘title element’ of each hierarchy followed by an equals sign.

Here is an example

Categories:

Markets =
  1 North America
    1.1 Canada
    1.2 United States
  2 European Union
    2.1 United Kingdom
    2.2 France

Products =
  1 Standard
  2 Advanced

3. Define reports and business drivers. In the course of preparing a spreadsheet for presentation or financial analysis, one may generally want to present the data within separate ‘reports’. For example, there may be a report which is a profit and loss statement, a report detailing return on assets, and a third report presenting cash flow analysis. Sunsight Modeller™ 1.0 enables one to divide business driver variables amongst such reports (note that a variable may appear in several reports). Each report may have its variables outlined according to the category hierarchies optionally defined above. Sunsight Modeller™ 1.0 will also automatically separate variables requiring data input from variables that are calculated from the data in other variables.

If categories have been defined and one wishes to have the report on the spreadsheet appear with any or all of the category elements in the outline, the next line should have the words ‘Breakdown by’ appear in it, followed by the titles of the category hierarchies to be included, which each hierarchy title separated by a comma. For example, a category breakdown might be written as ‘Breakdown by Products, Markets’. After that, the formulae are written directly, with each formula separated from the next one by a new-line. Formulae in Sunsight Modeller™ 1.0 must have the assignment variable appearing to the left of the equal sign.
4.2 Sample Business Algebra Model

Each period is one year.
The number of periods is 3.
The first period starts on 2005.

Categories:
Markets =
  1 North America
    1.1 Canada
    1.2 United States
  2 European Union
    2.1 United Kingdom
    2.2 France
Products =
  1 Standard
  2 Advanced

Report: Profit And Loss
Breakdown by Markets
#===================================================
Gross Profit = Turnover – Cost of Sales
Operating Profit = Gross Profit – Selling and Administrative Expenses
Profit Before Taxes = Operating Profit + Other Income – Interest
Profit = Profit Before Taxes – Taxes
Cost of Goods Sold = Labour + Raw Materials
Selling and Administrative Expenses = Selling and Distributions + Administrative Expenses

Report: Liquidity Analysis
Breakdown by Markets
#===================================================
Current Ratio  = Current Assets / Current Liabilities
Cash Ratio = (Cash + Short Term Investments)/Current Liabilities
Operating Cash Flow Ratio = Cash Flow from Operations / Current Liabilities

4.3 Sample Generated Spreadsheet

The sample Business Algebra Model shown above in section 1.5 generates, by way of Sunsight Modeller™, a full spreadsheet. We present here two screen captures of the automatically generated Excel spreadsheet derived from that sample:
The first sheet demonstrates the automatic separation of data assumptions variables from calculated variables (the numerical data in this sheet were entered manually, but all the cells not coloured in yellow were computer generated):

<table>
<thead>
<tr>
<th>Assumptions for Markets</th>
<th>Years</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>2005</td>
</tr>
<tr>
<td>North America</td>
<td>0</td>
</tr>
<tr>
<td>European Union</td>
<td></td>
</tr>
<tr>
<td>United Kingdom</td>
<td></td>
</tr>
<tr>
<td>Cost of Sales</td>
<td>27,095</td>
</tr>
<tr>
<td>Turnover</td>
<td>51,514</td>
</tr>
<tr>
<td>Selling and Distributions</td>
<td>12,805</td>
</tr>
<tr>
<td>Administrative Expenses</td>
<td>6,868</td>
</tr>
<tr>
<td>Other Income</td>
<td>12</td>
</tr>
<tr>
<td>Interest</td>
<td>1,846</td>
</tr>
<tr>
<td>Taxes</td>
<td>343</td>
</tr>
<tr>
<td>Raw Materials</td>
<td>19,324</td>
</tr>
<tr>
<td>Labour</td>
<td>7,171</td>
</tr>
<tr>
<td>Current Assets</td>
<td>13,401</td>
</tr>
<tr>
<td>Current Liabilities</td>
<td>13,955</td>
</tr>
<tr>
<td>Short Term Investments</td>
<td>1,226</td>
</tr>
<tr>
<td>Cash</td>
<td>1,678</td>
</tr>
<tr>
<td>Cash Flow from Operations</td>
<td>7,497</td>
</tr>
<tr>
<td>All Markets</td>
<td></td>
</tr>
</tbody>
</table>
The second sheet presents the calculations of profit and loss. Everything appearing in this sheet was automatically generated by the software, with no manual human input – indeed, the cells are security-locked to prevent tampering. The reader is invited to note the readability of the formula generated for the highlighted cell, as compared to a formula depending upon alphanumeric cell-references:

5 CONCLUSION

We have shown that consideration of spreadsheet user-interaction paradigms that differ from the standard ‘cell-matrix’ paradigm that has dominated the electronic spreadsheet over the past two and a half decades can provide significant benefits in multiple categories for business-orientated spreadsheet users. We have also suggested a list of desired features for a new ‘affordance paradigm’ for spreadsheet creation, and presented a working software system built upon the principles of such an alternative paradigm.
Further studies researching and quantifying how different spreadsheet user-interaction paradigms can impact usability, ease-of-use, error-rates, precision, readability, comprehension, maintenance, security and ease of auditing of spreadsheets are clearly called for. It also remains to be seen to what extent a new spreadsheet paradigm can overcome barriers to widespread adoption amongst users who have gotten used to the existing paradigm for decades. This latter challenge will likely require a combination of utility value and capitalising on network effects by integrating as readily as possible to already widespread tools and programs.

6 REFERENCES


Controlling the Information Flow in Spreadsheets

Vipin Samar, Sangeeta Patni
Extensio Software, Inc.,
19763 Merritt Dr. Cupertino, CA, USA
{vsamar, spatni}@extensio.com, 1.408.961.6050

ABSTRACT

There is no denying that spreadsheets have become critical for all operational processes including financial reporting, budgeting, forecasting, and analysis. Microsoft® Excel has essentially become a scratch pad and a data browser that can quickly be put to use for information gathering and decision-making. However, there is little control in how data comes into Excel, and how it gets updated. The information supply chain feeding into Excel remains ad hoc and without any centralized IT control. This paper discusses some of the pitfalls of the data collection and maintenance process in Excel. It then suggests service-oriented architecture (SOA) based information gathering and control techniques to ameliorate the pitfalls of this scratch pad while improving the integrity of data, boosting the productivity of the business users, and building controls to satisfy the requirements of Section 404 of the Sarbanes-Oxley Act.

1 INTRODUCTION

There are plenty of examples of full-fledged Microsoft® Excel based business applications with multiple inputs, outputs, and graphs that are being used for critical business processes in many companies. The reasons for Excel’s widespread adoption by its 150 million business users include its perceived simplicity, familiarity, and modeling abilities. However, the businesses also have to live with the various data collection and logic errors that creep into these spreadsheets. For the purpose of this paper, we would be focusing mainly on the errors in data collection and in the flow of information in and out of Excel.

While primitive data gathering tools that import external data into Microsoft Excel have been available from the inception, most users still depend upon populating their handpicked data into Excel using Control-C and Control-V. Given the nature of critical operations that the business users perform with Excel, and the increased control measures mandated by the Sarbanes-Oxley Act, it is increasingly becoming urgent for businesses to remove dependency on Control-C and Control-V primitives and also address the quality, freshness, and the accuracy of data in Excel.

This paper starts with providing some historical perspective of how users have traditionally managed the data-entry process for Excel. We then discuss some of the common pitfalls of getting data into the Excel spreadsheet, followed by how section 404 of Sarbanes-Oxley exacerbates the problems.

To address the problems associated with the information supply chain of Excel, we propose a Service-Oriented Architecture (SOA) based model coupled with an Excel Add-In. We describe how users can automate data collection through the use of Information Services instead of copying data, and discuss how this boosts productivity, improves data integrity, and addresses some of the requirements of Sarbanes-Oxley Act without compromising the flexibility and ease-of-use of Excel.

Copyright Extensio Software, Inc., 2005
2 MANAGING INFORMATION INSIDE EXCEL SPREADSHEET

Spreadsheets started off as a personal productivity tool for calculations and managing operations with persistent data. It then evolved into a scratch pad to keep lists, perform mathematical functions, and show graphs. Later, Excel began to be used for collecting data from multiple places, building information models, and conducting iterative and incremental analysis. Somewhere along this time, Excel also started getting used as a group productivity tool even though there were no specific features that helped with the collaborative effort. Today, Excel based applications exist in most companies performing tasks ranging from simple ones such as weekly reporting to complex ones such as financial accounting, budgeting, forecasting, and operational planning. Despite huge investments in IT, most business users still depend upon handpicking data for Excel from CRM, ERP, Portals, and databases.

In a survey by CFO IT [Durfee 2004] of 168 finance executives on the use of IT by corporate finance departments, they found that only 2 out of a list of 14 finance-specific technologies were widely used: spreadsheets (100%) and basic budgeting and planning systems (66%). The survey also reported that when executives were asked about the usage of spreadsheets five years from now, 91% of them thought that the spreadsheets would have the same or more importance.

The extent of Excel usage in enterprises is indeed quite deep and widespread. According to Forrester’s Keith Gile [Gile 2005], “14 percent of end users are producers - those who create analytic reports and author enterprise reports. The remaining 86 percent are consumers of the information and data.” He further adds that most business users (25% of the total) and casual users (30% of the total) prefer canned reports, or reports in Excel formats that they can then parameterize and use. Further, the extended enterprise users (38%) need read-only Excel reports.

As Excel is here to stay for a long time within businesses, it is important to mitigate some of the pitfalls without sacrificing its ease-of-use. Excel is indeed a free-format scratch pad, but unfortunately is getting used by business users who are not trained in structured programming, and version control [Panko 2000]. The free world of spreadsheets makes them vulnerable to the following types of errors:

- **Input errors**: These errors are due to inaccurate cut-paste, inadvertent changes in cells, incorrect links, import of incorrect data or import with wrong parameters.
- **Logic errors**: These errors are due to incorrect formulas, and incorrect input data.
- **Usage errors**: These errors include incorrect use of functions, ranges, and references.

For the remainder of this document, we would be concerned mainly about input errors, how data comes into Excel, and how it is maintained in the spreadsheet.
2.1 Current Methods for Getting Data into Excel

Here are the drawbacks of the common ways/tools used by users to get data in Excel:

<table>
<thead>
<tr>
<th>Tools Used</th>
<th>Limitations</th>
</tr>
</thead>
<tbody>
<tr>
<td>Copy-paste external data with Control-C, Control-V</td>
<td>• Manual process, with high error and no validation</td>
</tr>
<tr>
<td></td>
<td>• No trace of the source of data</td>
</tr>
<tr>
<td></td>
<td>• Users need to reformat the data as per their requirements</td>
</tr>
<tr>
<td></td>
<td>• Neither systematic nor reproducible process</td>
</tr>
<tr>
<td>Import the .xls or .csv file created by the application</td>
<td>• No linkage with the originating application</td>
</tr>
<tr>
<td></td>
<td>• Users need to reformat the data as per their requirements</td>
</tr>
<tr>
<td></td>
<td>• Does not support update of data using Excel</td>
</tr>
<tr>
<td></td>
<td>• Limited use when data comes from multiple applications</td>
</tr>
<tr>
<td>Import from databases using ODBC</td>
<td>• Considered complex for typical business users as it requires knowledge of database structures and SQL</td>
</tr>
<tr>
<td></td>
<td>• Not applicable when the source is not a database</td>
</tr>
<tr>
<td></td>
<td>• Practical only with few resources and few users</td>
</tr>
<tr>
<td></td>
<td>• No meta-data available for advanced manipulations</td>
</tr>
<tr>
<td>Import using web query</td>
<td>• Typically used for HTML reports from public pages</td>
</tr>
<tr>
<td></td>
<td>• Requires users to specify the complete URL along with parameters and embedded authentication information.</td>
</tr>
<tr>
<td></td>
<td>• No ability to select fields, or header information.</td>
</tr>
<tr>
<td>Import data using web services (Excel 2003 Prof. edition only)</td>
<td>• Considered complex for business users as they need to specify the web service, and map the XML schema</td>
</tr>
<tr>
<td></td>
<td>• Works only with web service resources</td>
</tr>
<tr>
<td></td>
<td>• Authentication issues with multiple web services</td>
</tr>
<tr>
<td>Access data warehouse with Excel Add-in</td>
<td>• Works only against historical data in data warehouse</td>
</tr>
<tr>
<td></td>
<td>• Not suitable for daily operational needs of business users</td>
</tr>
<tr>
<td></td>
<td>• Not possible to update data</td>
</tr>
</tbody>
</table>

None of the above approaches are satisfactory as they have the following deficiencies:

- **Loss of Linkage.** Once data is copied without tracking the source of data, any updates in the source cannot be propagated to the eco-system.
- **No Support for Multiple Sources.** Much of the data needed comes from multiple types of sources. Doing it manually is error-prone and time consuming.
- **Stale Data.** Without easy ways to refresh data, decisions are made with old data if it takes a long time to collect data.
- **Non-repeatable and difficult-to-manage process:** Each user performs these operations independently on their own spreadsheets making it impossible for IT to verify, automate, audit or validate the data transfer mechanism. Further, due to lack of IT management and control, efforts spent in building expensive data imports/exports cannot be re-used within the organization.
- **Read-Only Data.** Without being able to update the back-end right from Excel, users have to learn new applications, and depend upon copy-paste to transfer data.
- **Lack of Ease-of-Use.** If the data collection process is cumbersome, the users are likely to go back to their current cut-paste model to meet their objectives.
- **Missing Audit Trail.** Without the means to track the spreadsheets as they change, compliance and auditing becomes difficult.
2.2 Sarbanes-Oxley (SOX) Enters the Picture

SOX mandates that companies need some controls over all aspects of financial reporting. Most data that business users touch have some impact, whether direct or indirect, into the financial reporting of the business. Starting from a sales person, to an inventory manager, or a warehouse manager, all submit their figures using Excel. The channel or the resellers also aggregate their bookings, returns, and forecasts, all using Excel.

Computer World [Horowitz 2004] reported, “Fannie Mae made a $1.2 billion accounting error last year because of what it called "honest mistakes made in a spreadsheet. TransAlta Corp. took a $24 million charge after a bidding snafu caused by a cut-and-paste error in an Excel spreadsheet.”

The real problem is that the Control-C, Control-V takes almost zero time to master, and while this works in the favor of busy business users, it doesn’t create a systematic and reproducible process that can stand the scrutiny of their departmental policies and financial auditors. This increases the long-term cost of the Excel-based solution including its maintenance, debugging, auditing and compliance.

Excel’s autonomy is what made it popular, but without adequate controls, this can become a huge liability. The challenge is to address the following SOX compliance issues while keeping usability into consideration:

- Control over the data input and output process for spreadsheets
- Control over who is authorized to get that data
- Control over who modified the data, and when
- Control over how repeatable is this entire process

2.3 Paying for the Inefficient Information Supply-Chain for Excel

The key constituencies that pay for these inherent Excel weaknesses are:

1. **Business Users**: The users pay dearly by the hours they take in data hunting-and-gathering. They also take hours in learning how to get data from multiple sources.

2. **IT Department**: The IT departments pay because once the data leaves their home repositories they lose control over it. As the data spreads through the enterprise, there is loss of the attached metadata, security, management, integrity, and data constraints. Locking the data repositories from user access is not a viable alternative, as it would force IT to do all the reporting work, and increase their load.

3. **Business**: The business suffers because decisions are made with stale, inaccurate, and insufficient data. There is no single version of truth across the enterprise.

3 PUTTING CONTROLS ON THE INFORMATION FLOW INTO EXCEL

This section proposes Information Delivery with a Service-Oriented-Architecture (SOA) based server and an Excel add-in for controlling the flow of information in/out of Excel.

There are two root causes for the information flow problem. The first is that the users deal with data directly and touch it by hand, and the second is that there is no control over the data once it leaves the IT managed application repositories and reaches Excel. Also, we believe that if there is any complexity, it should be on the IT side and not the user side, because IT is more equipped and trained (and paid) for this complexity. If the data can be hidden behind an “information service” that the user can pick and use, the user would not have to know the various details on how to get the data. Further, since services would be built, designed and managed by IT, they will provide better data and access controls.
3.1 Introduction to Information Services

Information Services, broadly speaking, can be defined as a logically grouped set of information elements, extracted from data source(s) about an information entity (the “key”). An information service may have information elements combined from multiple data source, or could be the result of some transform operation on the original data. For example, a Customer Information service may take a customer ID and return the Customer name, address, phone number, and credit rating. End users view and consume this service without knowing that the customer name, address and phone numbers have come from their back-end CRM application, while the credit rating has come from their custom database application or a SOAP service. These information services are typically hosted on a server within the data-center.

By moving to a service-based approach, the users just specify the service they want (e.g. customer rating service), and they get the related data from the service directly into their Excel cells, without having to know the where, the how, and the when of the data. It is the service now that knows the details of getting the data and not the user.

3.2 Introduction to Service-oriented Architecture (SOA)

An information server using service-oriented architecture enables a loosely coupled integration of the back-end resources such that it masks the application consuming this data from the underlying IT complexity. This complexity includes the back-end resource structure, data-format, session management, security, connection pooling, and caching.

The main value of the SOA-based architecture is in deriving value out of the existing applications and resources without duplicating data, business logic or security efforts. The SOA based framework is typically composed of the following elements:

- Native interfaces to connect, authenticate, and access back-end sources. A SOA based system can access heterogeneous resources using SOAP or resource native interfaces such as SQL, HTTP, SAP BAPI, or custom protocol.
- Logic to combine and unify the data from multiple sources. This can be a simple JOIN statement to complex scripting depending upon the user requirements.
- Framework to expose the combined data elements as an information service that can then be consumed by end-users, devices or other applications.
- Framework to publish service directory and service schema

3.3 Introduction to Excel Add-in

Microsoft Excel provides well-defined means to add custom commands and features to Excel using Excel Add-ins. Examples include statistical and financial packages. While most Add-ins typically manipulate the Excel data, they can do lot more including contacting an external server and fetching external data. They can also define their own toolbar that can then be used in conjunction with the services offered by the Add-in.
3.4 SOA and Excel Add-In for Improving Information Flow Into Excel

The combination of a SOA based Information Delivery Server and an end-user interface coupling technology (Excel Add-in) at the user’s desktop provides the foundation for managing the information flow in and out of Excel. The underlying SOA architecture improves the accessibility and quality of data from back-end resources, while the Excel Add-in automates data collection, delivers data to Excel, and limits the human error.

Here are the main requirements of a SOA-based Information flow system for Excel:

- Automate data collection. Allow users to collect/refresh data without copy-paste
- Keep data connected. Enable one version of truth by linking cells to the source.
- Update data. Allow users to update data directly from Excel.
- Audit cell updates. Allow viewing of previous ‘n’ values for auditing or tracing.
- Version Control. Enables users to easily switch between versions of worksheets.
- Centralized control. Centrally manage all service/resource changes and security.
- Primed for growth. Support a large user base, both internal and external.

For a SOA based framework to provide data for end-user consumption, the technology needs the following additional components:

- Modules to map the result presentation layer to interfaces such as Excel
- Protocols to connect and authenticate various end-user interfaces
- Interface specific service publication and registry
- Interface specific update process

The SOA based information delivery server contains metadata about the services such as the data keys, connection details, transformation, assembly, resource information, and security. The service specific meta-data is stored in a XML rules database and used at run-time for request processing. If there are any changes to the query, they can be made without any user impact.

The server publishes a directory containing the services and attributes. The user chooses the service and input parameters, and sends the request to the server. The server responds by calling the enterprise.

Copyright Extensio Software, Inc., 2005
application using the meta-data, getting the result in the native format, extracting the relevant data, converting the data format and finally sending the data to the end-user.

The Excel Add-in provides the end-user interface and the delivery component. It sends the parameterized service request to the server using XML/HTTP. It then interprets the XML results sent back by the server and updates the cells. By sending the results back as XML, the Information Delivery Server retains the ability to send any other meta-data attached to the cell data. This allows for future extensibility. Examples include suggested refresh period, the associated update services for this element, and any special formatting.

### 3.5 Defining and Building Information Services

The IT department in conjunction with the business analysts and users typically develops the information services. Alternately, the application vendors can ship prepackaged information services that access their own applications, and then the IT department can create information services that combine data from those services. Once the services are defined, the business analysts can select the layout and the fields required by the users.

IT Developers take the following steps to build the services:

1. Identify information entities such as customer, including validation and mapping.
2. Identify the data sources that contain information elements for the entity.
3. Define the rules to fetch/update data from data store using the development tools. The rules can be written using resource specific wizards such as the SQL query wizard, SOAP wizard and the XML/HTML filter.
4. If required, define the rules for transformation, mapping, and assembly of data.
5. Define the presentation layer as appropriate for Excel, desktop, PDA or mobile.
6. Publish the information service on the services directory for the end-user.
7. Assign user/group permissions for information services.

Depending upon the complexity of the service, it may take from a few minutes to a few hours or more to build a set of information services, whether for lookup, reports, update, or alerts. SOA based architecture provides the required infrastructure for IT to serve, manage and control end-user access to information services.

### 3.6 SOA and Excel Add-In Integration Architecture

The overall architecture is shown below.
The Information Delivery Server and Excel Add-in work in tandem to manage the information flow into Excel spreadsheets:

<table>
<thead>
<tr>
<th>User Requirements</th>
<th>SOA Contribution</th>
<th>Excel Add-in Contribution</th>
</tr>
</thead>
<tbody>
<tr>
<td>Easily select data from multiple data stores, and refresh them later.</td>
<td>Provides information services containing elements spanning multiple data-sources.</td>
<td>Provides the ability to link and refresh Excel cells using information services.</td>
</tr>
<tr>
<td>Eliminate copy-paste, and use of SQL/XML to get data in Excel</td>
<td>Information services that hide the technical details of accessing data-sources</td>
<td>Provides menu based interface to get data directly within Excel</td>
</tr>
<tr>
<td>Update/Insert data in back-end from Excel</td>
<td>Provides the update interface if supported by applications</td>
<td>Consistent mechanism to specify and send updates</td>
</tr>
<tr>
<td>Have a replicable data collection process</td>
<td>Centrally available services allow reuse across users</td>
<td>Have persistent links to services in Excel</td>
</tr>
<tr>
<td>Maintain “single version of truth” and cell-integrity</td>
<td>Live on-demand connection to data-sources ensures latest data</td>
<td>Provides refresh on-demand, audit, version control and cell protection, for cell integrity</td>
</tr>
<tr>
<td>Ability to add internal and external data sources</td>
<td>Allows addition of information elements from new data sources with the tools</td>
<td>The newly published services are accessible from the service directory</td>
</tr>
</tbody>
</table>

3.7 SOA and Excel Add-In for Data Integrity, Audit, and SOX Compliance

SOA and the Excel Add-In help maintain the integrity and the freshness of the data. Since the data comes directly from the applications in a controlled manner, the integrity of the spreadsheet is not put in risk from copy-pasted data. Further, a direct refresh capability ensures that the financial reports have the latest data. SOA ensures that if there is any error in one of the input elements, it can be easily fixed and then refreshed by the dependent spreadsheets. SOA based Information Delivery Server can also provide centrally managed authorization control on per service-level for users/groups.

The Excel Add-In shows only the authorized services to users. It can also block the users from making any direct changes in the server-supplied data, while permitting for changes made with “refresh”. The Add-In can provide audit trail of the cells including the earlier values, time-of-update, and the user making the change. The user also can checkpoint the Excel spreadsheet and restore it to any of the previous checkpoints. This avoids the user having to keep manage multiple versions of the spreadsheets.

Thus, the Information Delivery Server in combination with an Excel Add-In provide controls for SOX compliance, and make the data collection and update process repeatable, verifiable and controllable.

4 BENEFITS OF SOA ENABLED INFORMATION DELIVERY FOR EXCEL

For any IT solution to succeed, it is important that it address the requirements of all major constituencies including the end-users, the business and the IT. The end-users on one end want agility, flexibility, and immediate delivery of data, and the IT on the other hand wants control, ease-of-deployment, and reuse of existing investments.
4.1 Benefits for the Business User

By moving away from a data-centric model to a service-specific model, the users get the following benefits without having to do any heavy lifting, or using custom tools:

- **Boosts productivity** by eliminating the time spent in moving data in/out of Excel.
- **Makes accurate decisions with latest data** as users can access fresh data.
- **Updates the back-end applications from Excel directly** instead of copy-paste.
- **Provides one version of truth** and reduce proliferation of spreadsheets.
- **Provides audit trail of changes** for tracking and debugging.
- **Reduces dependency on IT staff** for periodic reports as users have direct access.

4.2 Benefits for the IT Department

By moving to a SOA based information delivery, the IT benefits significantly:

- **Reduces risks of bad data** by delivering centrally managed Information Services
- **Shrinks the load of creating custom reports** by making users self-sufficient.
- **Connects internal and external applications** to Excel using the SOA model.
- **Support internal and external users** with support for HTTP/HTTPS
- **Reduces compliance issues** by putting easy-to-use controls for integrity.
- **Secures the Information Supply Chain** with centralized control and encryption.
- **Support Excel, desktop applications, and mobile** without any extra effort.

5 SUMMARY

Most businesses depend upon spreadsheets for financial reporting and managing operational processes. However, this flexibility comes at a significant cost of data integrity, loss of productivity, and loss of IT control. The paper highlighted some of the reasons for these information flow problems, and proposed a service-oriented architecture through which one can have adequate controls to mitigate these risks.

With a combination of service-oriented architecture (SOA) and Excel Add-In, the users can use information services instead of cutting-and-pasting data. This retains the link of the Excel with the enterprise data sources, and improves IT control over the data in the spreadsheet. Besides helping in compliance with Section 404 under Sarbanes-Oxley, the proposed approach boosts productivity, improves data integrity, and increases security while retaining Excel’s flexibility and ease-of-use.

REFERENCES


11 Developing an Auditing Protocol for Spreadsheet Models

Stephen G. Powell, Kenneth R. Baker, Barry Lawson

Tuck School of Business
Dartmouth College
Hanover NH 03755 USA
sgp@dartmouth.edu

ABSTRACT

We define auditing as the process of finding errors in completed spreadsheets. It is widely accepted that errors are prevalent in spreadsheets and that they can be extremely difficult to find. A number of audits of existing spreadsheets have been reported in the literature but few details have been given about how those audits were performed. Many different procedures have been recommended for auditing, including the use of test inputs, peer review, formula inspection, and sensitivity analysis of outputs. Auditing can also include the use of specialized spreadsheet add-ins. We report here a detailed, step-by-step auditing protocol we are developing and testing. The protocol uses two software tools (Spreadsheet Professional and XLAnalyst) to assist the auditor in locating potential errors. Our research is designed ultimately to provide insight into which auditing procedures, used in what sequence and combination, are most effective across a wide range of spreadsheet applications.

1 INTRODUCTION

Spreadsheets are widely used by end-users (non-professional programmers) to carry out an enormous variety of data and decision analysis tasks. While the appeal of spreadsheets as a computing platform lies in their ease of use, there is widespread evidence [Panko, 2000] that many spreadsheets in actual use contain errors. These errors can either be encoded in the spreadsheet itself (e.g., in an erroneous formula) or result from improper use of the spreadsheet (e.g., entering a value in a cell occupied by a formula).

The prevalence of errors in spreadsheets could be due to many causes. Some errors may simply be tolerated in spreadsheets because the spreadsheets themselves are not consequential. In other cases a spreadsheet may contain minor numerical errors but the decisions or insights drawn from it remain valid. Yet another cause may be that end-users build spreadsheets chaotically, without careful planning, and they have limited knowledge of how to audit their spreadsheets once they are built. Certainly, most spreadsheet users will attest to the difficulty of finding errors in spreadsheets once they are built. If the typical end-user does little error-checking because errors are difficult to find, then effective auditing methods and tools should be valuable.

For clarity, we define auditing here as the process of finding errors in completed spreadsheets. This is not to deny that a form of auditing is used during the construction of a spreadsheet. In fact, laboratory studies have shown [Brown and Gould, 1987] that end-users make and correct errors constantly while building spreadsheets. There is also a need...
for better understanding how end-users can reduce or eliminate errors in the design and construction of spreadsheets, but that topic is outside our scope.

There are many different types of audits and purposes for auditing a spreadsheet [Ayalew, Clermont, and Mittermeir, 2000]. An audit might be performed by a third party as part of a formal compliance review, or by a company insider for internal quality control purposes, or by the spreadsheet builder to validate the model before declaring it ready for use. The purpose of the audit might be to verify the accuracy of the numerical outputs to a high degree of precision, or to establish that the broad managerial conclusions suggested by the spreadsheet are dependable. Finally, the focus of the audit might be to verify that the spreadsheet accurately performs the calculations the builder intended, or to confirm that it helps solve a broader business problem. Given this wide variety of motivations for auditing spreadsheets, we can anticipate that different approaches would be effective in different circumstances. Auditing research should help establish which methods and tools are most effective in any particular circumstance.

Spreadsheet auditing has many similarities to auditing of traditional computer programs. Software engineers have developed and tested a variety of procedures for avoiding and finding errors, including reusable code modules, test data sets, and structured walkthroughs. One can hope that some of these methods would be of use to end-user spreadsheet programmers, but professional programmers and end-users are very different types of people and operate in very different environments. Professionals typically have more training in programming than end-users and more resources to devote to auditing. The challenge of developing auditing methods for end-users is to improve their use of the spreadsheet platform without destroying those features that make it appealing in the first place: simplicity, ease of use, personal control and freedom.

2 WHAT IS KNOWN ABOUT AUDITING

The literature on spreadsheet auditing is intermixed with that concerned with spreadsheet errors generally. There are three areas of research that are relevant to our work: field audits, code inspection experiments, and tests of auditing tools.

Panko cites seven reports on field audits of spreadsheets, but notes that most of these studies focus more on the errors found than on how the audits were conducted. Cragg and King [Cragg and King, 1993] inspected 20 operational spreadsheets. In these audits one person spent an average of two hours doing code inspection. Panko also reports on the audit of a large-scale capital budgeting spreadsheet at NYNEX. In this audit, teams of three each audited one of the six main modules of the spreadsheet. The audit included an explanation from the developer as to the logic of the module and how it related to the model as a whole. The team then verified formulas and checked cell references. One row in each column was studied in detail and the others were checked for consistency. Test data were used to audit some portions of the module. Finally, Excel’s formula auditing tool was used extensively.

Butler [Butler, 2000a, 2000b] describes an auditing procedure developed for HM Customs and Excise tax agency in the United Kingdom. This procedure involves the use of a software tool specially created for government auditing of small-business tax returns (SpACE). This auditing procedure has been documented in [HM Customs and Excise, 2001], which is by far the most detailed published auditing protocol available. Because the tax auditors are faced with thousands of spreadsheets to audit, the first goal of this
procedure is to identify the risks to the government posed by any one spreadsheet. Accordingly, the procedure involves a series of steps, at any one of which the auditor can terminate the audit. For example, the audit may be terminated if the likelihood of significant errors is judged to be low, or if the impact of errors is judged to be low, or if the resources required for a full audit are too high, and so on. Under this procedure, detailed code inspection is performed on a small subset of the original collection of spreadsheets.

When detailed inspection is warranted, the Customs and Excise procedure works as follows. First, the auditor identifies the chain of cells from inputs to the end result and uses the software to follow the chain of dependent cells so that the key formulas can be checked. Then the auditor checks the original formulas that were used to copy related formulas, and checks that the copies are correct. Again the software is used to speed up this task. Note that this procedure is fundamentally different from an exhaustive, cell-by-cell inspection. Butler (2000b) claims that this procedure saves considerable time while adding only a minimal risk that an error will be missed. Finally, fourteen types of high risk cells are checked for arithmetic and logical correctness. These include, for example, cells that contain constants, have no dependents, or involve complex functions such as NPV (net present value).

The second major line of related research involves code inspection experiments. Panko (2000) points out that explicit procedures have been developed and tested for code inspection of traditional computer code. To successfully find most errors, the number of lines of code being tested must be limited, as must the amount of code tested per time period. Even when these strict requirements are met, individual inspectors catch only half of the errors and inspection teams only catch about 80%. Only one similar study [Panko and Sprague, unpublished] is reported by Panko on spreadsheet errors, in which individuals found 67% and teams found 92% of all errors.

The third line of related research involves tests of the power of auditing software. Davis [Davis, 1996] reports on a study of two experimental tools, one an online flowchart tool and the other a visual dependency tracker. He shows that the two tools have about equal power to identify bugs, but does not compare them to not using a tool. Nixon and O’Hara [Nixon and O’Hara, 2001] compare the performance of five auditing tools in finding 17 errors seeded into a spreadsheet. The tools tested were Excel’s built-in auditing tools, Spreadsheet Detective, The Excel Auditor, Operis, and SpACE. The most successful tools were found to help in identifying over 80% of errors. The mechanisms that seemed most helpful to users were those that provided a visual understanding of the schema, or overall pattern, of the spreadsheet, and those that searched for potential error cells.

Our review of the literature suggests that very little careful research has been done on auditing protocols. Only two studies [Panko and Sprague, unpublished; Nixon and O’Hara, 2001] were found that tested the ability of auditors to find known errors when following a somewhat structured procedure. The only work of this type on auditing spreadsheets from the field is the work of Butler, and this is restricted to small business tax returns in the UK. The need for more research in this area is evident.
3 A PROPOSED PROTOCOL

We pointed out earlier that there are many different types of audits and purposes for auditing. To focus our discussion we will consider here just one well-defined type of audit. Imagine the situation of a single individual who is asked to audit a spreadsheet that was built by someone else. The auditor has no contact with the developer, just the spreadsheet itself, perhaps some auditing software, and limited time. The goals of the audit are to identify any outright errors in the spreadsheet and to note any potential errors or dangerous practices that might lead to errors when the spreadsheet is used. Our research is motivated by the desire to provide useful advice to an auditor in this situation.

Any auditing procedure should be both effective and efficient. Effective in this context means that the procedure identifies most or all of the errors or problems in the spreadsheet. Efficient means that the procedure gets results quickly and with reasonable effort on the part of the auditor.

Another important aspect of any auditing protocol is the sequence with which various tasks are undertaken. For example, research may show that testing extreme inputs and inspecting critical formulas are both effective procedures, but does it matter which one is done first? Some research has shown that auditing software can be useful in locating errors, but is it possible that use of more than one kind of auditing software would be more effective, or do all such tools identify essentially the same errors?

While it may never be possible to examine scientifically all combinations of approaches to auditing, research in this area should attempt to establish the effectiveness and efficiency of any proposed protocol. Therefore we have developed an example protocol, tested it under realistic conditions on a variety of spreadsheets taken from actual practice, and revised it based on lessons learned.

Our initial approach to defining an auditing protocol was to review the available auditing software. Roger Grinde provided us with a list of 45 software tools with some auditing capabilities. From this set we selected 10 tools to review in detail. Our final choice was to use two tools: XLAnalyst and Spreadsheet Professional. We selected XLAnalyst primarily because it is easy to use. It scans the spreadsheet and records whether any of 17 questionable conditions are met. It produces a report in the form of a spreadsheet that summarizes the potential errors found and it gives an overall risk score for the spreadsheet. Spreadsheet Professional was chosen because it appeared to be comprehensive, powerful, and well-supported. Both tools generate quantitative summary measures as part of their evaluation, and this feature seemed appropriate for a research investigation. Many of the software tools we reviewed include similar features. For example, most will produce a schema or map of the spreadsheet and many can display all the precedents of a single cell. Thus by selecting these tools we are not implying that they are more effective than other tools. A comprehensive comparison of all existing auditing software would be a massive undertaking in itself.

Our next step was to draft a step-wise procedure. At the same time, we developed a means for recording the results of the audit. (This aspect of our work was undertaken primarily for research purposes. We do not expect that an auditor in the field will necessarily need to devote time to documenting his work in this much detail.) Our procedure involves a mixture of quantitative and qualitative assessments. Here are the major steps in outline form with brief explanations (the complete protocol is given in an
Appendix; updated versions and the template we use to record the results are available at http://miba.tuck.dartmouth.edu/spreadsheet/index.html.

Preparation
- explore the workbook
- identify outputs and inputs

Preliminaries
- run XLAnalyst and Spreadsheet Professional’s Test Report tool

Purpose
- describe the purpose of the workbook
- describe each worksheet and record which sheets provide data to the given sheet

Performance
- test extreme inputs
- test the sensitivity of the output(s) to input(s)
- create a Tornado Chart
- record any error cells found in these tests

Map analysis
- examine the maps produced by Spreadsheet Professional for suspicious cells and record any error cells

Numerical data
- record the Overall Risk Rating and other summary measures from XLAnalyst
- record summary measures from Spreadsheet Professional

Functions used
- determine which Excel functions are used in each sheet

Potential errors
- record the cell address of any potential errors identified by XLAnalyst
- record the number of potential errors and the cell addresses identified by Spreadsheet Professional and investigate for actual errors

Design
- record how well the workbook follows good design principles

Documentation
- record how the workbook is documented

Security
- record security used in the workbook

Actual errors
- search for and record any remaining errors

4 SOME OBSERVATIONS

This protocol has been in development for several months and has been tested by five researchers. It continues to evolve as we test it against different spreadsheets. While we do not have definitive conclusions to report as yet, we can offer some general observations on our experience to date.

- The auditing software tools we are using produce large numbers of false positives. These are cell addresses that are flagged as errors when the cell in question is not actually an error. A common example is a growth rate formula such as
=D7*(1+$B$1)^2CS5.

This will be flagged by Spreadsheet Professional, for example, under “Numeric values in formula,” but surely this is not even a dangerous practice. The question for auditing is how to deal effectively with these false positives.

- While line-by-line code inspection is the only approved approach to serious auditing in traditional programming, checking every formula in a spreadsheet is tedious and may be impractical given time constraints. Our protocol hopes to improve on this approach by using performance testing, the maps created by Spreadsheet Professional, and the potential error reports from both XLAnalyst and Spreadsheet Professional. How efficient this approach is remains to be seen.

- Researchers have audited spreadsheets from the field, with an unknown number of errors, and tested spreadsheets in laboratory experiments. We are experimenting with both approaches. An open question is whether a protocol that is effective in finding seeded errors in test spreadsheets is also effective in spreadsheets from the field.

- A serious audit of a workbook consisting of 10-15 sheets, which is a very common size in the field, appears to take a minimum of 5 hours and can easily take several days. Can auditing time be reduced through practice? Are teams of auditors more effective than individuals, and if so how should a team audit be organized?

- Most spreadsheets from the field are essentially undocumented and poorly organized, making understanding and auditing difficult. What can we learn from auditing research that can improve the design of spreadsheets?

5 SUMMARY

In spreadsheets built by end-users, errors and dangerous practices appear to be common. Auditing a spreadsheet for these problems is an essential part of good spreadsheet engineering, but little research has been done to identify effective auditing practices. We have created and begun to test a detailed auditing protocol that uses two spreadsheet auditing add-ins (XLAnalyst and Spreadsheet Professional). Our protocol uses performance testing, schema maps, and potential error reports from the software tools, as well as detailed code inspection. We are currently refining the protocol and plan to test it with significant numbers of laboratory and field spreadsheets.

REFERENCES


Auditing Protocol 7.1

March 2, 2005

Note: This protocol assumes the user has available the Excel add-ins Spreadsheet Professional and XLAnalyst.

Preparation

The first step in auditing a workbook is to explore it in order to gain an initial understanding of its purpose and layout. This process will take between fifteen minutes and several hours, depending on the complexity of the spreadsheet. You may want to take notes while you explore the spreadsheet, but you should not focus too much on the details of any one sheet.

Since the purpose of every spreadsheet is to calculate one or more outputs based on a set of inputs, one of the first things you should try to identify is the location of outputs, inputs, and calculations. Look for clues in the sheet names or in labels on individual sheets.

Once you think you have located the outputs, try to get a general sense of how the outputs are determined. Use Trace Precedents in the Formula Auditing toolbar, or Display All Formulas (Control+`). (You can also find references to other sheets when in the Display All Formulas view by using Edit - Find and searching for the string “!”.)

Eventually you will want to determine the purpose of every sheet in the workbook and every major module in the most important sheets. Using Display All Formulas is helpful here, as it can show you which sheets are used to calculate the given sheet. Using Trace Precedents repeatedly on an output cell can show the pattern of calculations used to determine it.

Once you have a good overview of the entire workbook, you are in a position to record specific information about it. For each spreadsheet we audit we will record information on the following ten topics:

- purpose
- performance
map analysis
numerical data
functions used
potential errors
design
documentation
security
actual errors

What are errors?
An actual error exists whenever the numerical value in a cell is incorrect. Actual errors are distinct from potential errors which involve cells that could produce errors in some circumstances, but which do not currently produce incorrect values. An example of a potential error would be an IF formula that evaluates correctly with the given inputs, but incorrectly with different inputs. Another example of a potential error would be embedding a numerical constant in a set of formulas. This could lead to an error if some but not all of the constants were changed in use.

Errors are mainly due to mistakes in formulas, although occasionally errors involve text or formatting mistakes. The types of formula errors you should look for include:

- references to wrong cells
- incorrect logic
- parameters not referenced in all appropriate formulas
- formulas that evaluate to #ERROR or some other error condition
- IF-statements that do not evaluate correctly for different inputs
- LOOKUP-statements based on non-sorted data

Preliminaries
1. Run XLAnalyst by selecting XLAnalyst - Analyse This Workbook. XLAnalyst generates a separate workbook containing test results in a worksheet titled XLAnalystReport.

2. Run Spreadsheet Professional by selecting from the toolbar Test - Spreadsheet Test Reports tool. (Note: if you see a warning related to merged cells, click on Yes.) Under Sheets, check All Sheets. Under Include, check Summary, Range Names, Maps, General Tests, and Calculation Tests. Click on Options, go to the Test Options tab, and make sure all the boxes are checked. Then click OK twice. Spreadsheet Professional will run the Test Report tool and record the results in a new workbook. Note that Spreadsheet Professional analyzes each sheet in the subject workbook and for each sheet creates related sheets called Sheet Name Map and Sheet Name - Errors.
3. Create a single, integrated workbook by merging the template worksheet, the XLAnalyst sheet, and the Spreadsheet Professional workbook. Start with the Spreadsheet Professional workbook. This will include a Summary sheet, a Range names sheet, a series of Maps sheets (one for each sheet in the test workbook), a General Tests sheet, Errors sheets (one for each sheet in the test workbook), and an Additional notes sheet.

Copy the XLAnalyst sheet to the workbook after the Range names sheet. Name this sheet XLAnalystReport.

Copy each sheet of the test workbook before the corresponding Maps sheet, and move the Errors sheet for this worksheet to follow the Maps sheet. Once you are done you should have three sheets in order for each worksheet in the test workbook: the sheet itself followed by the corresponding Maps and Errors sheets.

**Purpose**

1. Open the template workbook Audit Report Template.xls. Rename it with the name of the workbook you are auditing, followed by “Audit Report.” You will record your audit information here as you work and save it as a sheet named Audit Report in the original workbook when you are done.

2. Write a short description of the purpose of the workbook (cell C3).

3. For multi-sheet workbooks, enter a sequence number for each sheet in row 1, starting in cell D1. Enter the name of each sheet in row 2, starting in cell D2. Enter a short description of the purpose of each sheet in row 3, starting in cell D3. Enter the sequence numbers of the sheets that provide data directly to the given sheet in row 4, starting in cell D4. Note: enter all the required information for the first sheet in column D, then proceed to enter the information for the second sheet in column E, and so on.

**Performance**

1. One effective way to understand the behavior of a spreadsheet is to test it with various inputs. This may uncover flaws in its design and ultimately may lead you to specific error cells. Some specific tests include:

   - testing extreme inputs
     - try large or small values for individual inputs, including negative values; try unlikely combinations of values
   - using Data Sensitivity
     - perform one-way or two-way sensitivity analysis using the Data Sensitivity tool from the Sensitivity Toolkit (http://mba.tuck.dartmouth.edu/toolkit/); look for unexpected behaviors
   - using Tornado Chart from the Sensitivity Toolkit
- vary each input one-at-a time and create a Tornado Chart (also available in the Sensitivity Toolkit); look for inputs that have no effect on the output or an unexpectedly large or small impact.

If you identify an error, record the cell address under Errors starting in B128, the type of error in C128 (for example, wrong cell reference), and how it was discovered (input testing, Data Sensitivity, or Tornado Chart) in D128.

**Map analysis**
1. Examine the Map sheet for each sheet in turn. Look for patterns that suggest potential errors. For example, you might see an F (for Function) followed to the right by several "<" signs (indicating the function has been copied to the right). If another F appears in the same row, that might indicate an error. Mark any suspicious cells by coloring them yellow. Then examine the corresponding cell in the test worksheet to determine if it is in fact an error. If it is, color it red and record it in the Audit Report Template under Errors starting in row 128). Record the cell address under Errors starting in B128, the type of error in C128 (for example, wrong cell reference), and how it was discovered (map analysis) in D128.

**Numerical data**
1. Record the Overall Risk Rating (Cell H6 in the XLAnalystReport) in cell C7 in the Audit Report sheet.

2. Copy the eight numbers from L33:L40 of the XLAnalystReport to the Audit Report sheet starting in cell C8 and ending in cell C15. Do not change the cell formats.

3. Copy the worksheet names from row 2 of the Audit Report (starting in D2) to row 17 (starting in D17). Under each sheet name, copy the six numbers between B4 and B10 on the relevant Errors sheet into rows 18-23 in the appropriate column. (Note that there is one blank cell in the source range, and this is ignored). Check that the row totals are calculated in cells C19-C23.

**Functions used**
1. Determine which functions (if any) are used in each sheet. For each sheet, use Go To – Special – Formulas to highlight all formulas. Then use Display All Formulas (Control+Shift+F7) to display the formulas. Record the names of the functions you find in rows 26-30 under these categories:

   - **Math (e.g., SUM,EXP)**
   - **Statistical (e.g., AVERAGE,STDEV)**
   - **Database (e.g., DCOUNT,VLOOKUP)**
   - **Logical (e.g., IF,AND)**
   - **Financial (e.g., NPV,PMT)**

**Potential Errors**
1. Return to the XLAnalystReport. Column K lists 17 possible error conditions (excluding Measures and System messages) in rows 10-30. These same possible errors are listed in the Audit Report sheet in B34:B50. Wherever you see "Found" in column K
of the XLAnalyst Report, record the cell address (without the sheet name) from column M under the appropriate sheet name in the Audit Report sheet, starting in column D, row 34.

Check these cells addresses and if any are actual errors record them under Errors starting in row 128. Under How discovered (column D) enter XLAnalyst.

2. Return to the Spreadsheet Professional Spreadsheet Test Report. Go to the Errors report for each sheet. Starting in row 12, these sheets report the number and cell locations of up to 25 types of potential errors. (All 25 potential errors are listed in the Audit Report sheet in rows 52-76 and 78-102.) The names of the potential errors are listed starting in cell A14, and the frequencies (number of offending cells) in cell B14. Spreadsheet Professional lists only potential errors identified on the current sheet, not all 25 potential error types. Thus the lengths of these lists will vary from sheet to sheet. Spreadsheet Professional also reports the cell addresses of all cells that violate a certain error type. This information can be found farther down in column A. The name of a given error type appears first in bold type and centered. Then a description of the error follows, which occupies two or three rows. Finally, the cell addresses of the offending cells are listed.

First, record the frequency of each of the potential errors that appear under Potential errors summary starting in row 12 in the Errors sheet. You will have to copy these numbers one at a time into the appropriate cell in the Audit Reports sheet in rows 52 to 76. First find the column that corresponds to the relevant sheet. Then find the row that corresponds to the appropriate type of potential error, and enter the frequency.

Second, record the cell address(es) for each of the potential errors as given in the Errors sheet. You will have to copy these cell addresses one at a time into the appropriate cell in the Audit Reports sheet in rows 78 to 102. First find the column that corresponds to the relevant sheet. Then find the row that corresponds to the appropriate type of potential error, and enter the cell addresses.

Working with one sheet at a time, spot-check each set of potential error cell addresses in the test worksheet. If any are actual errors, record them under Errors starting in row 128. Under How discovered (column D) enter SS Pro.

**Design**
Rate the workbook as a whole on the following seven criteria. Use a scale of 1 to 5, where 1 means “least positive” and 5 means “most positive.” Start in cell C105.

- Overall ease of understanding
- Use of modules
- Use of parameterization
- Use of range names for parameters
- Use of range names for formulas
- Ease of use
- Ease of communication
11 Developing an Auditing Protocol for Spreadsheet Models  
Stephen G. Powell, Kenneth R. Baker, Barry Lawson

Documentation  
Note whether these features are observable in the workbook. Use Yes/Some/No as possible responses. Start in cell C114.

- Model assumptions  
- Sources for inputs  
- Guide to sheets  
- Cell comments  
- Pseudocode for formulas  
- Notes in cells

Security  
Note whether these features are observable in the workbook. Use Yes/Some/No as possible responses. Start in cell C122.

- Protected cells or sheets  
  (Note: Select Tools – Protection and check each of the four options listed for any type of protection.)  
- Hidden cells  
  (Note: You can find this in the XLA AnalystReport, cell L20)  
- Data validation  
  (Note: You can use Go To – Special – Data Validation - All to highlights the cells in a worksheet that use Data Validation.)

Errors  
The final step in an audit is to identify any remaining errors in the workbook. You will want to use the Trace Precedents tool in the Formula Auditing toolbar and the Display All Formulas (Control+?) option extensively in your search for errors.

Ultimately, you must examine every formula in every sheet, at least quickly, in order to be confident that you have found all the errors in the workbook.

List any cells that contain errors starting in cell B128. Under How discovered enter Code inspection.
12 The use of spreadsheets to enable Sarbanes-Oxley 404 compliance.

Adrian Carter
DIRECTOR
SMARTTECH CONSULTING SERVICES LTD
http://www.smarttechs.com

ABSTRACT

Sarbanes-Oxley 404 (sox404) compliance is predicated on how the controls of data and information are used within an organisation so that the method of data interpretation is clear and may be reproduced. This will involve several new business orientated disciplines.

One of the most immediate requirements is from the prolific use of spreadsheets which are commonly used as a business tool for calculations, real time data capture and monitoring which is part of the business reporting submissions from financial systems. The compliance requirements the industry faces are a mixture of both procedural controls and technical solutions both have there place within a compliance remediation framework but neither should not be viewed in isolation.

Sarbanes-Oxley 404 a regulation that changes how companies treat financial reporting:

Sarbanes-Oxley 404 compliance is gaining momentum; the regulation represents a major change in the attitude to the control of data and information. Financial departments are moving from defining their requirements and evaluating their current controls towards implementation of remediation solutions. Financial software vendors have had time to build compliance controls into their offerings. Audit firms are gaining expertise in what works and what doesn’t. Some companies have met the initial deadlines and been declared as having their internal controls in order.

Relief and dangers from delayed compliance deadline:

However many companies are struggling to understand what is needed to meet the regulations; hence the recent US Securities and Exchange Commission (SEC) extension comes as a welcome relief, giving them more time to work through their processes, not to mention the time required to evaluate, implement, and audit their internal controls. The danger here is that companies relax their remediation programs; it is typical that when more time is given to tackle a large project, the extra time is largely spent in planning instead of implementation, not recognizing that the work generally takes longer than expected.

This is a familiar situation within IT projects where requirements are not fully defined and custom work is required to build or integrate systems.

Sarbanes-Oxley 404 compliance in predicated on how the controls of data and information are used within an organisation so that the method of data interpretation is clear and may be reproduced, this will involve several new business orientated disciplines.
Other Industries have gone through it before with other similar regulations:

Industries, like Pharmaceutical and Food, are coping the best, as they have been through this exercise before with similar regulations. The U.S. Food and Drug Administration’s regulation Title 21 Code of Federal Regulations part 11 Electronic Records; Electronic Signatures (21 CFR Part 11) and various guidance documents on current good practice (cGxP) for computerized systems have required these companies to go through the exercise of risk assessments, life-cycle development strategies, and implementation of controls that closely parallel the kinds of controls that Section 404 compliance requires.

Compliance controls can be split into two categories Procedural and Technical:

Procedural Controls rely on people following stated business rules, such as a Standard Operating Procedure or Quality Procedure. Technical Controls rely on software or hardware enforcing those business rules. Finding the balance between these two categories is a key factor to a successful implementation.

It becomes easy for a company to fall into the trap of relying solely on Procedural Controls. The cost of writing up a procedure and training the staff is seen as negligible in comparison to a technical solution. Procedural Controls give management some feeling of assurance, but as it is the Chief Executive Officer or Chief Financial Officer that will go to jail or be fined if these procedures are not followed, is it enough to have only Procedural Controls?

Consider that someone who is intent on committing fraud or breaking the law is not going to be terribly concerned about following the company rules.

Technical Controls have their place within a compliance remediation framework, ensuring that business rules are followed. However these also cannot be viewed in isolation as the requirements the industry faces will not be solved by new IT solutions alone. Technical Controls become both cost-prohibitive and practically unfeasible as the scope of the control grows. Additionally, there is often a gap between the control system and the actual process. This is a typical issue with new IT systems where the programmers, who are often the users, do not know what the detailed requirements are or how the new process will work in reality and often vendors do not help by not explaining that their solution requires a lot more than the initial offering.

Spreadsheets are one of the largest concerns:

One of the most immediate concerns within affected industries is a compliance solution for the prolific use of spreadsheets. Uses range from balance sheet calculations, to real-time systems and depositories for business reporting from financial systems.

The initial and most obvious areas of concern are the lack of security and lack of a robust audit trail. However as companies start to evaluate Excel and develop a set of requirements, it becomes obvious that there are serious deficiencies that show up their lack of Validation & Verification. Even if a traditional risk-based approach is used, trying to reduce the number of spreadsheets affected, they will find only a long list of issues and long faces trying to solve them.

Some companies have tried to eliminate or drastically reduce spreadsheet use by moving to databases or larger financial tracking systems, however, once the data is “released” from the “secure systems” either for reporting or further development the chain of custody is broken and the information can no longer be considered trusted data.
Additionally, spreadsheets containing macros often become problematic, only working when used by particular people, and there are always certain spreadsheets that departments just can’t seem to do without or

Others are pinning their hopes on upgrading to Excel 2003, with promises of better protection schemes and improved collaboration features. However we have seen time and again organizations going through an evaluation only to find that Excel still comes up short. While there has been some improvement, there are still some inherent gaps in functionality that do not meet requirements. Even with Excel 2003, further remediation of Excel is required.

**Remediation:**

The success of a remediation effort is predicated on how well the controls are implemented. The organization must consider each of the following prior to beginning:

What is needed is the introduction of as few new or replacement systems as possible and where a new system is necessary using solutions which can integrate with existing systems and infrastructure without reducing the flexibility spreadsheets provide.

**Planning the Remediation for spreadsheets:**

Like any project to be successful, the remediation effort will require good planning and management for both the documentation standardization and project implementation. This becomes readily apparent when considering the number of spreadsheets in use, the tasks that follow and the international dependencies of multinational corporations.

**Some of the fundamental steps:**

- Taking an inventory of all spreadsheets and detailing items such as the owner, users, purpose, linkages, compliance exposure, and use of formulas and macros.
- Detailing a Risk Assessment methodology and defining acceptable levels of risk. This includes setting probabilities and dollar amounts to such things as errors in spreadsheet formulas or data entry, non-compliance leading to material findings, and fraud.
- Performing a Risk Assessment on each spreadsheet to categorize the level of compliance risk or financial risk to the company.
- Performing a Gap Analysis on the general Excel to identify where Excel does not meet the SOX-404 requirements.
- Developing a Validation Plan for Excel and spreadsheets. This covers what is expected for verification that the spreadsheets meet requirements both for the high-level requirements and the particular spreadsheet’s requirements. The format of such documentation should be defined and standardized.
- Detail a high-level requirements document for Excel in general. This requires an understanding of the regulatory requirements including external influences and standards, such as
  - Good Financial Practices (GFP) espoused by Public Company Accounting Oversight Board (PCAOB),
  - Control Objectives for Information and Related Technologies (COBIT),
Committee of Sponsoring Organizations of the Treadway Commission (COSO) parts 16 and 18.

Additionally, a practical understanding of the capabilities and limitations of Excel in regard to the Authenticity and Integrity of data are required.

- Developing Requirements for individual spreadsheets where formulas or macros are involved. (It may be useful to categorize what level of effort will be expended based on the complexity of the macros.)

- Implementing and validating the remediation solution on both Excel in general and individual spreadsheets.

**Staff procedures and Training**

Developing and implementing Standard Operating Procedures and a Change Control policy and Corrective actions and Preventive actions (CAPA) systems so that changes to procedural or technical controls are reviewed, approved, revalidated and staff retrained.

Staff Training and Motivation Organizations must consider how changes will impact the employees. The utilization of existing systems can reduce the cost of training, increase staff acceptance, and be more easily verified that they will do the job as intended. This allows the organization to leverage a regulatory requirement into a business benefit and not just a financial cost, as well as increase management confidence that the system will be actually used by the staff.

**Spreadsheet solution requirements:**

The requirement companies are typically asking us for are solutions that provide Sarbanes-Oxley compliance without becoming prohibitively expensive solutions.

- Use existing spreadsheets with the ability to import current spreadsheets into the secure environment with minimal modification.
- Avoiding the additional concerns of using specific customisation of Excel, changing the existing systems or having to look for an alternative to Excel.
- Ability to continue to use the convenience and common functionality of Excel in a compliant regulatory environment.
- Read only file format for use outside the controlling environment
- A standard off the shelf software solution that provides full compliance, removing the need for individual spreadsheet Visual Basic Application (VBA) customisation.
- Provide the authenticity and integrity of data that is applicable for good information security practises (ISO/IEC 17799).
- Digital signature to enable data integrity even when emailed or used outside of the software.
- Electronic signature for recording, reviewing and approving data
- Logical security features that control access to and also to each individual spreadsheet.
- A fully secure independent, searchable, audit trail recording who, when and what changes are made, which is kept within the spreadsheet removing the need for the separation of the data for auditing storage, backup and retrieval.
• Integration, interfacing and embedding with other applications that use spreadsheets e.g. Electronic Document Management System or databases or business systems.
• Easily deployed and implemented by either internal IT resources or outsource companies.

Conclusion:
Companies wish to continue to use the flexibility that spreadsheets give their business and do not want to invest in expensive “fixes” which restrict the use of one of the best tools businesses utilise. One example of a software solution that meets these requirements is Wimmer Systems DaCSTM for Microsoft® Excel.

The list seems daunting, but many companies, and other industries, have gone through this exercise using risk based streamlining, covering both procedural and technical solutions to reduce the implementation time and impact on the organization. Separating the general requirements for Excel from specific requirements for particular spreadsheets allows for the use of templates. Standardization and the use of automated tools for tasks such as setting spreadsheet formats, inventory and documentation storage increases user acceptance. The exercise is made less painful and looked on with a greater air of optimization by continuing with the use of spreadsheets and not requiring the migration of the common spreadsheet functionality into different systems.

Adrian Carter
Director
Smarttech Consulting Services Ltd
Smarttech Consulting Services, Ltd is a privately owned company established to meet the demand for enabling technology for compliance. SmarttechCS consults, distributes, and supports software solutions which provide knowledge from data, value from quality, and financial return from compliance. The company is headquartered in Stow-in-the-Wold, Cotswolds, United Kingdom. More information on SmarttechCS services can be found at http://www.smarttechcs.com. +44 (0) 1451 870637

Since 1977 Adrian has utilized his knowledge of most aspects of regulatory compliance within the Pharmaceutical industry interacting with the client at director level. Since qualifying as a Graduate of the Royal Institute of Chemistry, this has involved working within a pharmaceutical analytical laboratory, an instrumentation supplier, providing software solutions and over last few years, as a senior manager, using a consultative process to recommend consultancy led technology enabled solutions for regulatory compliance with an acceptable business case for a return on investment.

DaCSTM is a trade mark of Wimmer Systems Inc
Microsoft® Excel is a registered trademark of Microsoft Corp.
13 Why, How and When Spreadsheet Tests Should be Used

J C Nash and Jody Goldberg

J C Nash, School of Management, University of Ottawa  nashjc@uottawa.ca
Jody Goldberg, Maintainer, Gnumeric, jody@gnumeric.org

ABSTRACT

EuSpRIG, as a Risks interest group, is concerned with the various things that can go wrong when spreadsheets are used to perform important calculations. To detect and measure differences or errors, tests are devised. This paper considers why users of spreadsheet calculations, administrators of organizations that create or use spreadsheets, and developers of spreadsheet software may all wish to run tests. We also suggest some ways that the large volume of detail work may be sensibly organized and automated. The timing of tests is also discussed. Our work is motivated by developers trying to build better spreadsheet processors who need to test new releases to ensure compliance with design goals or to improve the programs. However, the same tests can be used to compare different programs or versions. In this latter use of our tests, we believe the primary concern is consistency of and confidence in results. Suitable tests can reveal changes from expected behaviour, and help the user distinguish changes in the processor from those in the spreadsheet content.

1 BACKGROUND – WHY SHOULD WE TEST?

Over the history of spreadsheets, there have been many complaints about errors in the programs that are used to build and run spreadsheets themselves. EuSpRIG authors and related web sites are well-represented in any list of references on this subject, and we are loath to single out particular papers.

Errors in the programs are less likely to result in erroneous calculations, however, than errors by the user-builders of the spreadsheets, as eminently illustrated by “The Wall” case study of Ray Panko (Irons, 2003; Panko, 2003). Indeed, it is argued (Grossman and Özlik, 2004, among others) that those interested in improving the quality of spreadsheets should work to make it easier or more likely that the user-builder will carry out his/her tasks correctly. This can be accomplished by a combination of improvements to the spreadsheet processor, better training, and the use of tools and procedures. Here we will use “builder” to refer to those working with the spreadsheet and “programmer” or “developer” for those developing the processor software.

Some tools for improving spreadsheets look for situations that indicate common errors (e.g., XLSior, www.xlsior.com) and have sometimes been termed “audit” tools. This can be confused with the provision of an audit trail of the work done by the spreadsheet builder. That functionality can be provided for some spreadsheet processors. In particular, the authors are associated with the TellTable infrastructure for providing an audited and collaborative use of the OpenOffice calc spreadsheet processor and with work that is in
progress to add a much lower-level audit track to the Gnumeric spreadsheet processor. The overlap of these interests has led to the present endeavour, since the introduction of the audit tracking into the spreadsheet processor requires changes to the source code, and each change has a potential to alter the actions of the program.

The goal of this paper is to present the issues in constructing appropriate tests that allow for rapid verification or characterization of a spreadsheet processor, to propose some options for addressing these issues and to introduce some preliminary examples of implementations of the test ideas. While the initial design audience is the programming team, there are much wider applications of such testing, in particular for IT managers to verify that the spreadsheet software under their purview is consistent across users, meets or exceeds some level of “quality” for particular functionality, and has not been corrupted in some way.

2 CHALLENGES TO TESTING – HOW SHOULD WE TEST?

The type of testing that we propose to do is frequently called “regression testing” (Kaner et al., 1999; Orso et al., 2004). This is not to be confused with least-squares regression, a statistical method to estimate functional models for data. The essential idea is that one can run a sequence of tests and receive a “green light” if all the tests “pass”. There are two main technical concerns in applying the idea to spreadsheet program testing:

- Spreadsheets are generally loaded manually and their output is usually to the screen unless the user-operator intervenes. For ease of use, we want to output the results of tests in a graduated level of detail to a file that identifies, at minimum, the program and test data version/date as well as the run time and date. Furthermore, we want to be able to chain a sequence of test spreadsheets to allow an omnibus test to be constructed.

- For the spreadsheet programmer, many of the tests we want to run involve functions that take arguments and return results that are floating-point numbers. This complicates the task of the test constructor hugely, as special functions are notoriously difficult to compute to a given relative or absolute precision over the full range of possible inputs. Indeed, most testing will be carried out using a combination of “random” inputs as well as some inputs that may be likely in some circumstances to yield errors. For some functions such as calendar calculations and database and lookup functions, Monte-Carlo style testing is the only reasonable approach. However, in order to allow for strict comparability between separate test runs, we shall generate the “random” inputs just once and build them into the test spreadsheets.

A separate issue for spreadsheet testing arises from the market dominance of Microsoft Excel®. This is the socio-political need for “new” spreadsheet processors to give results equivalent to Excel. Of course, what we mean by “equivalent to” and by “Excel” are open to much discussion. Equivalence could mean bit-by-bit equality in output, though we argue that outputs should be equal to within some relative or absolute tolerance. For functions that are non-numeric, for example, some data lookup or sorting actions on data with ties in data ordering, a tolerance makes no sense, and we would like, but clearly cannot guarantee, algorithmic compatibility. We also believe that it is important to specify the version / build of the Excel processor when making comparisons. Nevertheless, one of our design objectives is to characterize the ways in which spreadsheet processors are different from each other, which includes differences from and
between particular versions of Excel. We believe that the characterization of the differences is likely of greater importance to IT managers than the raw “correctness” of individual functions or their collection in the spreadsheet processor. It is, of course, of considerable interest to us, wearing the hat of programmer, to provide users with the best functions we can supply. It is with some pride that Gnucric was able to improve some of the statistical functions acquired from the R project (http://www.r-project.org (1425 2005-2-22) an open source, high-performance statistical package and programming language) and to be able to give back the improved functions to the R team.

From the perspective of the IT manager who has a number of copies of Excel deployed, there is also the question of version equivalence. We believe suitable tests can be devised to allow differences in action to be revealed. However, determining these may require running tests on several machines. Windows operating systems do allow more than one version of Excel to be installed simultaneously, but “Microsoft does not recommend it.” The Microsoft support item http://support.microsoft.com/kb/214388 (1535 2005-2-25) gives instructions for installing two versions of Excel on one computer. These imply that the order of installation is crucial, which can be a considerable nuisance in practice. VMWare allows multiple instances of an operating system to be launched, thereby offering one work-around for such testing. Another option, that we are using quite happily, is the use of CodeWeavers Crossover to launch Excel under Linux, since this tool does not prevent different versions from being loaded.

3 AUTOMATING TESTS

The first stage in automating tests is the simplification and stratification of the results. That is, we want to perform many detailed tests, then provide summaries at various levels. As an example, consider testing a function that we shall call “SPRIG”, where we want our output to be of the form

\[
\text{No. of SPRIG tests that failed = } x \text{ out of a total of } y \text{ (at tolerance } 1.0E-6)\]

Clearly we want the number \( x \) to be zero. The central idea is to have a hierarchy of vectors of counts, with individual tests for specific arguments being the lowest level. The top level will be the aggregation of all the tests. Indeed, we shall want to prepare graphs of the outputs versus different tolerances used. However, there is an obvious decision to be made concerning the set of tests to be aggregated within a single spreadsheet file. That is, we can have multiple worksheets in a spreadsheet file, and these can contain the detailed tests. While it is possible to consider aggregating all our tests into one spreadsheet file, this will get too large for some systems to process. There are also some limitations on rows, columns and worksheets within some processors. For example the default row limit for OpenOffice calc is 32000 rows, but this can be changed as indicated in http://sc.openoffice.org/row-limit.html (1500 2005-2-25) Thus we may choose, for example, to group all our tests of trigonometric functions into one file. An obvious particularization of this idea is to put the summary information on the first worksheet of any file.

Clearly, we need to be able to load and execute a spreadsheet file, output the results to a file, then close the file. Such a facility must then be chained so a sequence of such test operations can be run and the results summarized. Furthermore, if the output is in a relatively well-defined format, we can use a “difference” tool (diff in Unix/Linux) to identify differences or changes between runs.
- Load and recalc:
  This can be accomplished by using a command-line interface to issue a command such as

  \path to MS Office\excel.exe test1.xls

  provided that manual recalculation is turned off. Clearly there are going to be different commands for different spreadsheet processors and operating systems.

- Output and close:
  The issue here is more tricky as we now need to execute some sequence of commands within the spreadsheet processor. The commands must force recalculation and then output part or all of the spreadsheet. We are able to do this for Gnumeric, are currently preparing a VBA autorun macro for Excel, and plan before the presentation of this paper to attempt an equivalent functionality for OpenOffice calc.

- Aggregation and comparison of results:
  We can consolidate several sequences of commands such as those above into a script to process several test spreadsheets, then process the results externally with another program. If the output format is based on text, Perl is an obvious candidate language for this, but any programming tool that handles text easily is suitable. Text is also a sensible choice for the more or less standard diff programs. Moreover, we can arrange the output to be convenient for viewing or analysis. For example, we could use HTML to provide the results as a Web page, or a csv (comma delimited text) file to allow it to be displayed and further processed in a spreadsheet.

To minimize the scripting work, we are attempting as far as possible to consolidate tests into as few spreadsheet files as is reasonable. This also minimizes the work of keeping track of the test files.

4 CONSTRUCTING TESTS

4.1 Logical, Integer and Programming Functions

The testing of functions with a finite number of results is certainly simpler than testing those that take floating-point arguments (we do not believe any common package still works in fixed-point). A major concern, however, is that all functions handle "invalid" arguments correctly. The current set of Gnumeric tests at [http://www.gnome.org/projects/gnumeric/function-info.shtml](http://www.gnome.org/projects/gnumeric/function-info.shtml) (1934-2005-2-24) is a strong base for development in this direction. It includes test spreadsheets for

- Bitwise Operations
- Address Functions
- Date and Time Functions
- Database Functions
- Information Functions
- Logical Functions
- Lookup Functions
- Textual Functions
- Operators
- Engineering Functions

158
Financial Functions

4.2 Floating Point Functions

For reasons quite separate from the focus of Eusprig, this is the main focus of the current test development work and we assume that it will be an ongoing activity as new functions are added to spreadsheet processors and as computers evolve to include new functionality that impacts the way in which elementary functions are handled. To indicate the way in which we plan to proceed, the Gnumeric test page already includes an initial attempt at a comprehensive test of trigonometric functions. The tests are of two main types,

- attempts to supply inappropriate inputs that the processor should detect and report
- comparisons of results that should evaluate to known “answers” that we can test.

These use suitable identities that should, under some models of the approximating algorithms, stress these functional approximations. We have based these on the work of Cody and Waite (1980), but are hoping to soon incorporate some ideas of Beebe (2004). Ideally, one would like the thoroughness and ingenuity of Kahan’s (1983) Paranoia, but its tools are essentially only for the arithmetic and not the functions. (Kahan was one of the primary motivators of the IEEE arithmetic standards that are the basis for most modern floating point hardware.)

We believe the testing activity for mathematical and statistical functions will continue for some time. Note that McCullough and Wilson (1999, 2002) have also developed some testing tools in this area.

4.3 Format of Test Files

All the test files in the Gnumeric suite are provided as Excel xls files. This has been done so that users outside the Gnumeric community can apply the tests to their own spreadsheet processors. We are also able to compare output with that from different versions of Excel, Quattro and OpenOffice (and possibly others) for purposes of compatibility assessment.

5 RESULTS

The progress of the activity that is the focus of this paper takes two forms:

1) The test spreadsheets that we are building and using to test the builds of the Gnumeric spreadsheet processor. These, as well as the source and binaries for Gnumeric can be downloaded via links on http://www.gnumeric.org. (1930 2005-2-24) There is also a test version of Gnumeric for Windows. Macintosh users can find a port of various Gnome tools, including Gnumeric on http://openosx.com/office/ (1550 2005-2-25).

2) The regression test execution script and report writer. These tools are undergoing rapid change. We will gladly share them with those interested in collaborating on testing spreadsheets, especially those willing to port them to Windows, Macintosh or other platforms. It is our intention to make them widely available as soon as they are stable enough that we will not receive too many “bug reports”.

159
A consequence of this work, we hope, will be the internalization of the audit trail capability within Gnumeric as well as functional capability that is as good as can be provided. Moreover, by making our activity and code open to review and contribution, a framework is provided for ongoing improvement. We offer our work to the spreadsheet user community as a service, as a challenge and as an invitation to contribute.

REFERENCES


Exploring Human Factors in Spreadsheet Development

Simon Thorne, David Ball
University of Wales Institute Cardiff
Sthorne@uwic.ac.uk Dball@uwic.ac.uk

ABSTRACT

In this paper we consider human factors and their impact on spreadsheet development in strategic decision-making. This paper brings forward research from many disciplines both directly related to spreadsheets and a broader spectrum from psychology to industrial processing. We investigate how human factors affect a simplified development cycle and what the potential consequences are.

1.0 INTRODUCTION

Human factors are present in every activity that humans undertake. Human factors really describe the frailties of interaction and interface between man and the world. In this paper we focus on how these factors affect spreadsheet modellers. The choice of spreadsheets is not arbitrary, recent research (Fernandez, 2002 and Gosling, 2003) has shown that organisations rely heavily on the use of spreadsheets to make strategic decisions and that many business critical processes are implemented using spreadsheet applications. This sort of reliance on spreadsheets is tactically dangerous, considering the issues that arise from Human Factor research.

Consider a simplistic development cycle consisting of: Plan, Build and Test as shown in figure 1. Throughout this cycle there are a number of different human factors that will impact on the quality and integrity of the model developed. This paper will explore the factors that effect the development cycle.

Simplistic development cycle

![Simplistic development cycle](Figure 1)
2.0 PLANNING STAGE

The effective planning of an information system is paramount to its success. In order to effectively plan an information system, one would have to be a trained information system professional. This leads us logically to examine the profile of the people developing spreadsheets and assess the skills they have in planning spreadsheet development.

2.1 SPREADSHEET DEVELOPERS

There is no typical spreadsheet developer in the modern business world. The reason for this is the great flexibility that spreadsheets offer, allowing a range of professionals to develop them. Most spreadsheet developers are end user developers by definition. End User Development (EUD) is the process of allowing end users to develop applications, using end user tools, to enhance business in some way. As end users, they will not necessarily be trained IS professionals. Indeed, many end users use computers as a necessity to perform their job satisfactorily, and EUD provides an enhancement to normal activities. EUD as an activity is paramount to information systems development and even software development. As end users, spreadsheet developers are not trained as software engineers and hence they have no knowledge of structured methodologies or processes that constitute software development. The consequences of developing software with no methodology came to a head in the 1980’s with the ‘Software Crisis’. There was a large upheaval of processes and standards by the industry to improve the quality of software using structured design techniques. Uptake of these methods has been widespread in the software industry. The same standards were not applied to EUD, although there was research published in the 1980’s that proposed frameworks for the management of EUD (Brown and Bostrom, 1989 Munro et al., 1987 and Alavi et al., 1987). As was observed by Gosling (2003), there has been little uptake on these management strategies and hence spreadsheet development is ad-hoc and chaotic. Figure 2 highlights this problem. The graph is the response to the question “Do you apply a methodology when developing your spreadsheet?"

![Figure 2 (Gosling, 2003)](image-url)

This is further reflected by the findings of a large-scale investigation into End User Computing in 34 UK organisations (Taylor et al., 1998) This study highlighted the lack of training given to End Users in formal information systems methodological approaches. Application of such management strategies, as Alavi et al. propose, is proven to improve the quality of the end product; the improvement in software quality after the software crisis is strong evidence of this as observed by Yourdon (1997). The fundamental problem is that the
developers of software are humans and humans make mistakes especially when they are not trained properly. In addition overconfidence plays a role. Overconfident modellers under time pressure feel no need to plan, since that would take time.

## 2.2 OVERCONFIDENCE

‘Often wrong but never doubting’ - Anon

Overconfidence in human activity is prolific; research shows that in relatively complex problems, humans are consistently overconfident. In particular Russo and Schoemaker (1992) examine the costs and causes of overconfidence in decision making. This overconfidence does not just apply to novice or inexperienced professionals. The same rules apply to ‘experts’ as Lusted (1977) and Oskamp (1965) demonstrated with physicians and clinical psychologists respectively.

Overconfidence, when seen from a spreadsheet development point view, is really concerned with all stages of the plan, build, and test development cycle. An overconfident spreadsheet developer will not plan, test or even question the validity of the work they have produced. This results in poorly designed, untested applications that are potentially full of errors. This practice is obviously risky in normal circumstances but consider more strategic applications and this practice becomes critical.

Research into spreadsheet modellers and overconfidence has shown that both individuals and groups demonstrate chronic overconfidence (Panko, 2003). Panko found that overconfidence ranged from 80% to 100% with the emphasis of overconfidence with individual modellers. Panko also extended his research to measure the effect of presenting the participants of the experiment with evidence that all spreadsheet developers were overconfident in their efforts. This resulted in a slight improvement in percentage accuracy of spreadsheet models developed and reduced the overconfidence of participants. This minor improvement after stark and blatant warning demonstrates the pervasiveness and severity of the problem. Burnett et al. (2003) also recognised overconfidence as a problem and demonstrated how software engineering principles could improve the accuracy and efficiency of spreadsheet developer models. Burnett’s experiment applied a testing methodology to a number of participant created spreadsheets. The results showed that the participants managed to correct 92-96% of errors found i.e. they audited their work and discovered errors, which they corrected. This is clearly an effective method of catching and correcting errors. What this study does not reveal is the number of errors that the participants did not detect, which, according to reported statistics could be between 80-100%. Burnett also provided an extensive testing methodology (Burnett et al., 2001) based upon software engineering techniques which improved the quality of spreadsheets by considering them in terms of executable and non-executable programs. Utilising this method, participants achieved between 0 and 31% (detected) error on a select number of problems.

The testing methodologies proposed by Burnett et al. are accurate and effective methods of catching errors. However, we return to the original question, if the end users are overconfident in the first place, will they consider the need for testing?

### 3.0 BUILD STAGE

At the build stage there is the greatest opportunity for human factors to impact on the quality and integrity of the spreadsheet. In this segment we explore several fundamental cognitive human issues that affect accuracy in development and even how we interact with spreadsheet programs.
3.1 HUMAN LEARNING AND MEMORY

As Gross (2001) observes, human learning is a hypothetical construct that cannot be observed directly but is implied in the improvement of cognitive and mechanical skills. According to Howe (1980), learning involves perception and memory. It is well accepted by the psychological community that learning and memory are essentially the same thing. The actual mechanical process of learning is based on principles of pattern matching and learning from experience, that is to say adjusting behaviour based upon previous experience to gain some more desirable result.

The ‘cognitive load’ (the demand of cognitive tools for a specific task) is also important. According to Kruck et al. (2003) the cognitive load is based upon four interlocking supersets: Skill Character; Working memory; Long-term memory and Task Demand. Within these supersets there are several subsets such as problem solving, memory load and accuracy. Assessing each subset in each superset allows one to build a picture of the cognitive load for a given task. Kruck et al. applied this method to a number of different everyday tasks that ranged from typing to routine medical diagnostics. He also applied this method to spreadsheet tasks, the results of this underlines the high cognitive demand on spreadsheet modellers, see table 2.

<table>
<thead>
<tr>
<th>Tasks</th>
<th>Skill Character</th>
<th>Working memory</th>
<th>Long term memory</th>
<th>Task demands</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Problem solving</td>
<td>Perceptual input</td>
<td>Planning</td>
<td>Unit task structure</td>
</tr>
<tr>
<td>Typing</td>
<td>Low</td>
<td>High</td>
<td>Low</td>
<td>Low</td>
</tr>
<tr>
<td>Driving a car</td>
<td>Low</td>
<td>High</td>
<td>Int.</td>
<td>Low</td>
</tr>
<tr>
<td>Mental multiplication</td>
<td>High</td>
<td>Low</td>
<td>Low</td>
<td>High</td>
</tr>
<tr>
<td>Balancing check book</td>
<td>High</td>
<td>Low</td>
<td>Int.</td>
<td>High</td>
</tr>
<tr>
<td>Writing a business letter</td>
<td>High</td>
<td>Low</td>
<td>High</td>
<td>High</td>
</tr>
<tr>
<td>CPA doing income tax</td>
<td>High</td>
<td>Low</td>
<td>High</td>
<td>High</td>
</tr>
<tr>
<td>Routine medical diagnostics</td>
<td>High</td>
<td>Low</td>
<td>High</td>
<td>High</td>
</tr>
<tr>
<td>Spreadsheet tasks</td>
<td>High</td>
<td>High</td>
<td>High</td>
<td>High</td>
</tr>
</tbody>
</table>

Table 2 (Kruck et al., 2003)

Kruck et al. were experimenting to determine if training spreadsheet users would affect their accuracy. Kruck concentrated on four elements of cognition that was labelled a framework for cognitive skills, see figure 3.

![Figure 3 (Kruck et al. 2003)](image-url)
It was found that the only element that improved significantly after training was logical deduction. Kruck et al. found that by improving the participant’s logical reasoning skills, the quality of the spreadsheets they developed was higher i.e. they had fewer errors.

From a cognitive point of view, developing spreadsheets is a method of developing software using syntax and logical constructs in much the same way writing a program in C++ or Visual Basic. In the latter two, the developer will have to learn the syntax and constructs over a period of time before the program will achieve what is required. Spreadsheets were designed to be a tool that could be utilised by non-information systems professionals. This has both advantages and disadvantages, while the user can develop a spreadsheet with no formal training, the probability of novice developers making mistakes is heightened. Consider a novice user modelling a spreadsheet; the interface is intuitive and logical and the use of complicated syntax is initially limited. Once some level of problem complexity is breached, the user will have to begin using more complicated statements. In spreadsheet applications, the syntax of arguments is often complicated and the flexible intuitive interface is replaced with a single line of code for a formula. Considering the user is not a software engineer or a programmer, this interface has serious implications on how effective the user will be at manipulating the environment to best suit their needs. Research has shown that a limited natural language interface impacts on the effective programming and learn-ability of an application (Napier et al., 1989). There is also a negative effect on the users programming ability, where the user has little knowledge of the commands involved in the environment (Napier et al., 1992). In particular the users will consistently make mistakes when developing spreadsheet applications beyond some level of complexity (Thorne et al. 2004). That is not to say only Novices make errors, indeed there is strong evidence to the contrary (Oskamp, 1965).

Human working memory is also an area that has been under investigated in EUD. The principles of ‘Miller’s threshold’ (Miller, 1956) state that when considering a problem, the subject will start to make errors after they are manipulating greater than 9 concepts simultaneously. If we consider this in terms of spreadsheet applications, these sort of complex and abstract formulae are common. Considering the problematic syntax and the abstract nature of programming formulae in spreadsheets, Miller’s concept danger limit is particularly important. Whilst it is difficult to interpret Millers use of concepts into the spreadsheet paradigm, one could view concepts on a cell-by-cell basis. Using that analogy, concepts would be elements of a formula in a cell. Considering the complex argument structure of spreadsheet applications, spreadsheet modellers must routinely breach Miller’s threshold. If we apply the principles of Halstead’s difficulty (Halstead, 1977) to a spreadsheet formula, the complexity becomes apparent as this method breaks down formulae into operands and operators thus providing us with the concepts used in an argument (Thorne et al., 2004).

Added to Miller is the work of Michie (Michie et al., 1989) who demonstrated the poor link between the pairing of human and machine strengths based upon an appreciation of human learning and cognitive processes. Michie argued that the human computer interaction was fundamentally limited due to the way in which humans interact with the computer. Michie essentially argued that the roles of machine and human in interaction did not exploit either’s strengths. His points are still relevant today, as the method in which we interact has not changed significantly since the paper was written.

### 3.2 HUMAN ERROR

Human error, unlike some of the other topics in this paper, is sourced from many different disciplines. Psychology initially started the interest but since there have been many disciplines interested in this phenomena. Reason (1990) produced the ‘Generic Error Modelling System’ (GEMS) based upon an understanding of human error taken from many disciplines. Reason proposed that errors are made on one of three levels: Knowledge based, Rule based or Skill
based. Rasmussen (1986) laid the foundations of this when investigating human error in industrial processing plants. Using these paradigms we can classify EUD error and thus target counter measures to manage EUD activities more effectively. Further, Fraser and Smith (1992) investigated the errors created when comparing human behaviour to the norms of probability, casual connection and logical deduction. This research yielded evidence of humans making mistakes in simple and repetitive tasks, known as Base Error Rate (BER). This concept states that regardless of the simplicity and repetitive nature of a task, there will always be a base level of error present. This phenomenon was observed in an experiment where participants were required to match colours with colour names correctly. Reason (1990) also found evidence of this phenomena and was brought to attention in EUD by Panko (1998). There are other numerous examples that include BER in spelling and grammar, calculation tasks, prediction and interpretation. Evidence gathered by Panko (2005) demonstrates a wide range of quantities, which vary significantly depending on the task. For example, a typical rate observed in spelling BER ranges from 0.5% to 2.4% (errors per word). Indeed Panko concludes that a reasonable estimate for BER in any simple activity is 0.5%. In comparison more complex tasks such as programming yield a BER of around 5%. This suggests a fairly a relationship between complexity of task and BER – the more demanding a task is, the higher the level of BER.

3.4 MODELS AND PARADIGMS OF HUMAN COMPUTER INTERACTION

Human Computer Interaction is a wide discipline that includes physical, biological and technical aspects of interaction between user and computer. There are several models of interaction that exist, the most popular being execution-evaluation cycles, Norman (1988), this method breaks interaction into seven stages. These stages describe the sequential process of a user planning, implementing and evaluating their work. This model represents the process of interaction between the computer and user, when the user has a specific task in mind, i.e. Print out the report. It can however, be applied to EUD, since the user will follow the model to produce a spreadsheet or a database using approximately the same steps. In examining the stages, it is suggested that some of the processes are not followed in EUD. The lack or misinterpretation of certain stages, i.e. evaluating the system state in respect to the goals and intentions is one of the causes of poor quality in EUD systems. Indeed, the above statement is open to all kinds of interpretation, whether it is the bias, as described by Fraser and Smith (2003), that causes the user to incorrectly interpret results or the failure to adequately test the system due to a lack of knowledge of structured methodologies.

An alternative way of interpreting human computer interaction, even spreadsheet development, is centred on problem driven modelling. Put simply, the user utilises the computer and their own cognitive faculties to solve a problem. The spreadsheet application is essentially the implementation of the cognitive model developed beforehand by the user. Problem solving in humans can be viewed as problem space searching as Newell and Simon (1972) suggested. Problems state space searching is the process of forming a goal state (what the user wants to create) a current state (the point that the user currently resides at) and the valid operators to change the current state to the goal state. The goal state in this context could be general or specific. It could be to create a spreadsheet that represents a business problem or more specifically the sum of two cells in a spreadsheet to produce a total. Consider the latter example, the goal state is a formula that sums two cells; the current state is nil (there is no part of the formula produced). The valid operators could be mathematical symbols (+ - / *), cell names and addresses (C1, B1 etc) and the applications specific operators (SUM). In this instance the problem space allows more than one valid goal state, there are several ways of writing a formula that will sum two cells. It is now at the users discretion to decide on the goal state that they desire. Selecting the best goal state presents the user with some significant problems. How does the user decide which is the best solution to the problem or are they even aware that there are other valid goal states. This paradigm of
problem solving is utilised in machine learning techniques since this model of problem solving lends itself to the area. When machines are presented with multiple goal states, the machine will assess according to efficiency, this may take the form of the goal state that is the most compact or requires the least processing power. For a human to assess the goal states in the same way would be problematic. In this simple example the human could make the decision but in larger problems where there may be tens or hundreds of valid goal states, it would take the user a significant amount of time to resolve to the best solution. It is this kind of problem that humans are weak at, evaluating large amounts of data in terms of efficiency, which contains large amounts of replication; a computer on the other hand is naturally good at this.

When we consider the way in which a user interacts with a computer, in light of problem space searching, to create an application that is a representation of a system, there are several fundamental processes. The first element is matching patterns in real–world examples and realising trends in those patterns that form some rule or judgement. The second is then manipulation of mathematics to represent that system accurately and lastly using logical deduction to classify the results accordingly. Now if we consider table 1, the natural strengths of the average human and the typical conventional computer some discrepancies arise.

<table>
<thead>
<tr>
<th></th>
<th>Pattern matching</th>
<th>Generating real-world examples</th>
<th>Manipulating mathematics</th>
<th>Logical deduction</th>
</tr>
</thead>
<tbody>
<tr>
<td>Human</td>
<td>Y</td>
<td>Y</td>
<td>?</td>
<td>?</td>
</tr>
<tr>
<td>Computer</td>
<td>N</td>
<td>N</td>
<td>Y</td>
<td>Y</td>
</tr>
</tbody>
</table>

Table 1

From this table we can deduce that humans are strong at generating real-world examples and pattern matching but weak at mathematical manipulation and logical deduction. Conversely, computers are strong at manipulating mathematics and logical deduction but weak at generating real-world examples and pattern matching. If we then apply this to EUD and spreadsheets in particular we can see that the current paradigm places strain on the natural weaknesses of the human and doesn’t exploit the computers full potential. Table 2 shows the current paradigm in spreadsheet development.

<table>
<thead>
<tr>
<th></th>
<th>Producing formulae</th>
<th>Generating real world examples</th>
</tr>
</thead>
<tbody>
<tr>
<td>Human</td>
<td>Weak</td>
<td>Strong</td>
</tr>
<tr>
<td>Computer</td>
<td>Strong</td>
<td>Weak</td>
</tr>
</tbody>
</table>

Table 2

As can be seen in table 2, the human is charged with providing the computer with the formulae, at which they are naturally weak. The computer then uses the formulae in the spreadsheet but does not exploit the massive potential that it has in terms of mathematics; it merely calculates data. A new paradigm that exploits the merits of both the user and computer would allow greater interaction. Ideally a method that would play on the strengths of both human and computer would improve the way in which the two interact. One such alternative novel solution would require the human to produce examples of attribute classifications and the machine would then deduce the function of those examples and generalise to new unseen examples. This approach has been coined ‘Example Driven Modelling’ (EDM) (Thorne *et al.*, 2004) which uses machine learning techniques to produce a more accurate system of creating
representative systems. Machine learning, in the context of EDM, is best described as the ability to adapt and extrapolate patterns in data as defined by Russel and Norvig (2003). To be more specific, the particular branch of machine learning that interests the researcher is Neural Networks and their use in example attribute classifications of data. For example, the user provides simple examples of the problem data. This data is then fed into the learning machine and it produces an equivalent model of the problem. Thorne et al. (2003) discussed an experiment to test the relative levels of accuracy gained from both traditionally modelling a formulae and utilising an EDM approach, over successively more difficult written problems. The results of this study found that producing the formulae with the traditional method was error prone (80% of models with error). The results of the EDM method yielded a much lower error rate (2% of models with error). These findings are reinforced by Michie et al. (1989) who compared human and machine learning over a series of experiments. A learning machine and human were given information regarding the legality of Rook – King moves in Chess. Both participants were given the same information and then Human and Machine learning was determined and compared. This experimentation revealed that machine learning can be more effective and efficient than human learning; the machine was consistently more accurate making better use of the information made available to it. Much of Michie’s work throughout the 1970’s to 90’s was concerned with Machine Learning Techniques (MLT) and the comparison of those techniques with equivalent human abilities (Michie, 1979 and Michie, 1990). He also revealed insights into the human learning process through his work, which he tried to represent through MLT (Michie, 1982). By summarising Michie’s work, the general goal of his research was to exploit human learning concepts via symbolic artificial systems to provide some machine or method that could learn more effectively. Michie found that machine learning was highly accurate, when compared to human learning, but that it was often too specialised. The machines could only ever perform a small number of tasks satisfactorily and beyond their domain, there were useless. In contrasts humans have greater generalised skill than specialised skill, affording them ‘graceful degradation’ in skills and knowledge. It is perceived that a novel approach such as EDM could greatly improve accuracy by delegating much of the work to the computer rather than the user.

4.0 TESTING STAGE

The testing stage is the final point before the user decides that the model they have produced is adequate for the task it was designed for. In addition to lack of formal testing methodologies that is implied since most spreadsheet developers aren’t IS professionals, bias is considered in testing.

Gilovitch et al. (2002) considers the heuristic methods and bias implicit in everyday life. The most relevant parts of this text refer to the bias present in seemingly objective judgements. For example, the trend towards predicting an outcome favourably due to the fact that the subject has a vested interest in the outcome (Armor and Taylor, 2000). Further, Fraser and Smith (1992) also discusses the concept of hypothesis fixation where once a hypothesis is formed about a decision, the subject will misinterpret the results to show that their hypothesis was correct. Fraser also examines confirmation bias, which is closely linked with hypothesis fixation. Confirmation bias is the tendency for individuals to test their own work with conditions that favour a positive response. Consider the flexibility and variable accuracy of End User Developed spreadsheet applications and the above factors become critical. The issue of bias becomes critical once the user is required to evaluate their work. Pryor (2004) theorised that End Users test their spreadsheets by using the ‘sniff’ test. That is to say, if the figures roughly match what they are expecting then the spreadsheet is valid. If we consider hypothesis fixation, confirmation bias (Fraser and Smith, 2002) or optimistic bias (Armor and Taylor, 2002) in this context, the little testing that is applied can be rendered invalid.
5.0 CONCLUSIONS

Although this paper has considered the fundamental issues in Human Factors, it is not exhaustive. There may be other equally important factors that this paper has not covered that impact on spreadsheet development.

If we re-visit the original Plan Build Test analogy (see section 1.0), we can demonstrate at which stage the particular human factors play a significant role. Figure 3 shows the modified diagram.

<table>
<thead>
<tr>
<th>Development cycle</th>
<th>Threats to Integrity</th>
</tr>
</thead>
<tbody>
<tr>
<td>Plan</td>
<td>Lack of structured development methodologies knowledge</td>
</tr>
<tr>
<td>Build</td>
<td>State Space Searching, HWM, Cognitive load</td>
</tr>
<tr>
<td>Test</td>
<td>Hypothesis Fixation, Optimism Bias, Confirmation Bias.</td>
</tr>
</tbody>
</table>

Figure 3

This diagram shows the particular issues that arise at each stage of development. For Example the greatest threat to integrity at the plan stage is the lack of structured development methodologies knowledge.

In addition to the stage specific threats, there are overarching issues that discretely affect each stage. The two overarching factors are Overconfidence and Base Error Rate. For example a modeller may be overconfident in planning their model, they may spend a minimal amount of time constructing a plan if at all. The same applies to testing; they may test their model inadequately due to the fact that they are confident that their model is accurate, indeed the plan stage may well be dismissed altogether if the modeller is overconfident. Base Error Rate (BER) plays a similar role, a user will be pre disposed to BER whilst they are building the spreadsheet but also when testing it.

In conclusion, Human Factors play a significant role throughout the spreadsheet development cycle. Further, Human Factors should be investigated further to allow the spreadsheet-modelling world to build applications and processes that are sympathetic to such issues.
Contents

References


Miller, G. A. (1956), 'The magical number seven, plus or minus two', Psychological review, 63, pp 81-97.


Oskamp, (1965), 'overconfidence in case-study judgements', The journal of consulting psychology, 29, pp 261-265


Panko, R. (2003), 'Reducing overconfidence in spreadsheet development', EUSPRIG Building better spreadsheets from the ad-hoc to the quality engineered, pp 49-57, 1-86166-199-1

Panko, R. (2005), 'Basic Error Rates', Internet http://panko.cba.hawaii.edu/HumanErr/Index.htm, Available, accessed on 10.6.05 12.34pm


Rasmussen, J., (1986), Information processing and Human-Machine interaction, an approach to cognitive engineering', North Holland series in system science and engineering, 12, 0-444-00987-6


Excelsior: Bringing the Benefits of Modularisation to Excel

Jocelyn Paine

www.j-paine.org/ and www.spreadsheet-factory.com/

ABSTRACT

Excel lacks features for modular design. Had it such features, as do most programming languages, they would save time, avoid unneeded programming, make mistakes less likely, make code-control easier, help organisations adopt a uniform house style, and open business opportunities in buying and selling spreadsheet modules. I present Excelsior, a system for bringing these benefits to Excel.

1 INTRODUCTION

There are two ways of constructing a software design. One is to make it so simple that there are obviously no deficiencies; the other is to make it so complicated that there are no obvious deficiencies. The first method is far more difficult. [Hoare, 1981]

The only way to write complex software that won't fall on its face is to build it out of simple modules connected by well-defined interfaces, so that most problems are local and you can have some hope of fixing or optimizing a part without breaking the whole. [Raymond, 2003]

You can model the value of modularity in a decision-theoretic way by using the Black-Scholes equation. Modularity increases your set of options for modifying a system, and it's a basic result of options theory that having a portfolio of individual options is more valuable than having an option on a portfolio. (Intuitively, a program whose components you can selectively upgrade is more valuable than one which is a monolith which must be upgraded in a big bang.) But to use that result, it must actually be the case that your modules are modular - and that's where you need abstraction as a semantic property. [Krishnaswami, 2005]

If you want to avoid having a missed bug affect an arbitrarily large proportion of the system, the answer is compartmentalization (more modularity), with effective enforcement of the boundaries between compartments. These mechanisms are able to prevent a missed bug from affecting an arbitrarily large proportion of the system. [Sitaker, 1998]

Spreadsheets lack the most fundamental mechanism that we use to control complexity: the ability to define re-usable abstractions. They deny to end-user programmers the most powerful weapon in our armory. Can you imagine programming in C without procedures, however clever the editor’s copy-and-paste technology? [Peyton Jones et. al., 2003]
This paper describes research on providing Excel with features for modular design, providing the advantages and avoiding the complaint voiced above. Section 2 explains modular design and its benefits, and should interest spreadsheet users and managers not lucky enough to have encountered these already. Section 3 introduces the research plan and how it fits with the rest of the paper. Sections 4 and 5 describe a simple mathematical treatment of spreadsheets and its use for adding modularity, with examples. Section 6 describes Excelsior, a programming language based on these principles. Section 7 briefly explains how this work could be continued and fitted to Excel. Section 8 briefly explains how Excelsior relates to, and can be used from, the Prolog programming language. Section 10 describes four applications.

2 MODULAR DESIGN AND ITS BENEFITS

Modular design is making an object - be it car, television, disc drive, or spreadsheet - from parts or modules which can be understood each on its own, not needing knowledge of other parts with which we might join it. The idea is to think about modules' insides separately from their outsides. Speaking generally – this applies as much to, say, mechanical engineering as to software - a module makes a "contract" that it will, given specified inputs, generate specified outputs. All the user need know is where to connect the inputs and outputs, and how they are related. The rest of the module's workings are up to its implementor, as long as he or she honours the contract.

2.1 Top-down design - plans for understanding, writing, testing, and documenting spreadsheets

Top-down design means thinking in terms of a hierarchy of modules. When fault-finding a hi-fi, we don't break into the chips: we read their specifications, ensure we understand why these components are connected as they are, and test that each is in fact receiving the right inputs. Thus, the structure of something – including a spreadsheet - built from modules gives us a plan to follow when trying to understand it.

In the same way, top-down design gives us a plan to follow when writing, testing and documenting new ones. Code one module at a time; test each thoroughly, then treat as a reliable component; reuse modules wherever possible. Most programmers know this: I emphasise it because some with only spreadsheet experience will not.

2.2 Localisation of concerns - easy updating, less work, fewer mistakes, a uniform house style

When we want the same calculation in different parts of a program, or in different programs, we could code it anew each time. Or we could code it once as a module, then insert that wherever we need it. This is the localisation of concerns: all the code concerned with a particular calculation is in a single clearly defined location.

Localisation of concerns saves time: instead of constantly redesigning, recoding, retesting, and redocumenting some calculation, the programmer just inserts a module from a library. It reduces risks: the less code, the fewer mistakes. And it saves work when updating code. Suppose we have many programs, all needing the same calculation. And suppose we want to extend the range of inputs the calculation can handle, or we find in it a mistake. If all programs use the same module to do the calculation, that's the only thing we need fix. But if the programmer has coded the calculation anew for each program, they need fix it as many times as there are programs; finding all these programs becomes
difficult; new mistakes become more likely. Localisation of concerns also helps organisations achieve a uniform house style.

2.3 Interchangeability of parts and suppliers

If all that's important about a module is its specification and external connections, we can replace it by any other module having identical specification and external connections, even if the workings are different. This is interchangeability of parts. In spreadsheeting, we could replace one module by another that's faster or more precise; that has been enhanced to handle complex numbers or Euros; that is cheaper, has better customer support, or a vendor less likely to go bust.

2.4 Code control

Being able to divide spreadsheets into modules would also aid code control, since we can track mistakes and alterations to each part separately. However, most code-control systems want source code as text, not as spreadsheets. The software I describe here makes this possible by enabling spreadsheets and spreadsheet modules to be saved as text.

2.5 Business opportunities

Modules for spreadsheets would create business opportunities in selling spreadsheet parts which others can use. Conversely, businesses could save effort and reduce bugs by outsourcing to experts. It's like wiring up a mass of circuitry from scratch, versus buying tested and guaranteed circuit boards, each with predefined connections, operating specifications, guarantee, and customer support.

3 RESEARCH PLAN

The research described here breaks down into the following steps:

1. Finding a mathematical representation for spreadsheets. This is explained in the next section.

2. Defining functions which act on objects so represented, combining and extracting parts or modules, and bearing in mind the points about modularity made in Sections 2 and 7. These functions are described in Section 5.

3. Designing a programming language to be as close as possible to these. This is Excelsior, explained in Section 6.

4. Excelsior is implemented in Prolog. There are sound reasons for not inventing yet another programming language when so many already exist. On the other hand, if we provide instead just a library for use from Prolog, we are bound by Prolog’s syntax and semantics, inconvenient for many users. This stage, therefore, looked at the pros and cons of implementing a Prolog interface to Excelsior, described in Section 8.

5. By now, Excelsior had almost become a practical language for handling spreadsheets, usable as a "scripting language" by programmers accustomed to such as Perl, Python, and Rexx. In this stage, therefore, were added features likely to increase its usefulness, such as formula editing. These are explained in Section 6.
6. The operators of Section 4.2 - can transform spreadsheets to be easier to understand. However, they need information about the spreadsheet author's intentions. Trying to discover these is the "structure discovery" of Section 4.6. In this stage, therefore, were added other features useful to structure discovery.

7. Excelsior is not for the typical Excel user; but it is a good foundation for a modularisation tool that could be used by them. Designing this, bearing in mind users’ skills and expectations, is the next stage of research. This is explained in Section 7.

4 SPREADSHEETS AS MATHEMATICAL OBJECTS

By treating spreadsheets as mathematical objects, this section defines operators for constructing new spreadsheet parts, splitting existing ones into parts, abstracting them, and joining them together. Unlike with other programming languages, these must take into account spreadsheets’ visual aspect - layout - as well as their content. Mathematicians may note that the work was inspired by category theory [Goguen, 1991] (useful in generalising the notion of “putting together”), sheaf semantics [Goguen, 1992], horizontal and vertical module composition [Goguen, 1996], and abstraction [Tennant, 1981].

4.1 Spreadsheets are sets of equations

Spreadsheets are, in essence, sets of equations. Here's an example; these equations could have come from a simple accounting spreadsheet:

A2 = 2000  
A3 = 2001  
B2 = 1492  
B3 = 1560  
C2 = 971  
C3 = 1803  
D2 = C2 - B2  
D3 = C3 - B3

4.2 Improving intelligibility by transforming equations

Transforming a spreadsheet's equations can make them clearer. Suppose for example that the author of the spreadsheet above meant column B to be expenses, C to be sales, and D to be profits. Rewriting the equations accordingly makes them easier to understand:

Year[2000] = 2000  
Year[2001] = 2001  
Sales[2000] = 971 etc.  
Layout Year[2000:] as A2 downwards  
Layout Expenses[2000:] as B2 downwards  
Layout Sales[2000:] as C2 downwards  
Layout Profit[2000:] as D2 downwards

4.3 Compiling spreadsheets from specifications

Transforming equations in the other direction lets us specify a spreadsheet's calculations separately from its layout, and in a way that uses meaningful identifiers and cell
groupings. For example, we could transform the second example to the first by replacing
Year[2000] by A2, Year[2001] by A3, and so on as directed by the layout statements. This is compiling equations to a spreadsheet.

4.4 Separating calculation from layout

Compiling is powerful, because by varying the layouts, we can map the same equations to a spreadsheet in many different ways. Thus we could compile the same three-dimensional table to vertically stacked 2-d cross-sections in one spreadsheet, to a horizontal sequence of cross-sections in another, and to one slice per worksheet in a third. It would also, for example, be equally easy to change the layouts so data runs from right to left, to accommodate Hebrew and Arabic readers.

4.5 Decompiling spreadsheets into specifications

The opposite to compiling is decompiling. This starts with the spreadsheet and rewrites its equations to be more intelligible, as in Section 4.2.

4.6 Structure discovery

Before we can decompile a spreadsheet, we need to know how its author meant its cells to be grouped. For example in the above example, cells A2 and A3 were meant to form a table of years. We also need sensible names for them. This - uncovering its author's intentions – is structure discovery.

As [Clermont, 2004] and [Hipfl, 2004] explain, we can devise heuristics to guess this implicit structure. To find out which cells belong in the same array, we can seek: regions surrounded by text or blank cells; sequences of identical formulae; ranges passed to SUM and other aggregating functions. To discover the array bounds, we can look for sequences of numbers such as years, and for sequences of common words such as month names. These often act as subscripts to the cells below or on their right. To guess names for the arrays, we can use labels in neighbouring cells. Because Excelsior is a complete programming language, any such heuristic can be coded in it; to simplify this, it provides spreadsheet grammar rules [Paine, 2004(a)].

5 PRIMITIVES FOR MODULARISING SPREADSHEETS

I shall use the baby accounting spreadsheet of Section 4.1 as a running example.

5.1 The □ operator: combining sets of equations

Suppose we want to put labels over the columns to say that column A is year, B expenses, C sales, and D profit. Let's write these labels as equations:

A1 = "Year"
B1 = "Expenses"
C1 = "Sales"
D1 = "Profit"

Then to insert the labels, we need only combine these equations with those defining the account spreadsheet. I shall do this, using the brackets ( and ) to enclose sets of equations, and □ to combine them. (□ forms the union of two sets under the constraint
that there cannot be two equations with the same left-hand side but different right-hand sides.). The spreadsheet we want, with accounts and labels, can then be written as:

\[
\begin{align*}
  &\{ A1 = "Year", B1 = "Expenses", C1 = "Sales", D1 = "Profit" \} \\
  &\{ A2 = 2000, A3 = 2001, B2 = 1492, B3 = 1560, \\
    &\quad C2 = 971, C3 = 1803, D2 = C2-B2, D3 = C3-B3 \}
\end{align*}
\]

We can insert calculations too. Suppose we wanted to calculate the tax - assume it's 33% - on profits and show the result in column E. We can write:

\[
\begin{align*}
  &\{ A2 = 2000, A3 = 2001, B2 = 1492, B3 = 1560, \\
    &\quad C2 = 971, C3 = 1803, D2 = C2-B2, D3 = C3-B3 \} \\
  &\{ E2 = D2\times0.33, E3 = D3\times0.33 \}
\end{align*}
\]

So □ lets us insert cells that are independent of existing cells (the labels), but also cells that depend on existing cells (these calculations).

5.2 The □ operator: shifting equations

Now suppose we want to insert a column of text on the left of column A. We need to shift existing columns right, which I’ll do with □. This takes a set of equations and an (X,Y) pair, and shifts the spreadsheet represented by the equations X places right and Y down:

\[
\begin{align*}
  &\{ D3 = C3-B3, D2 = C2-B2 \} □ (2,10) = \\
  &\{ F13 = D13-C13, E12 = D12-C12 \}
\end{align*}
\]

Then if (say) we wanted to insert a column of years as Roman numerals, we could write:

\[
\begin{align*}
  &\{ A2 = "MM" , A3 = "MMI" \} □ \\
  &\{ A2 = 2000, A3 = 2001, B2 = 1492, B3 = 1560, \\
    &\quad C2 = 971, C3 = 1803, D2 = C2-B2, D3 = C3-B3 \} □ (1,0)
\end{align*}
\]

The □ and □ operators therefore have two purposes which fit well together: joining parts of spreadsheets visually; and chaining together inputs to outputs.

5.3 The let: naming sets of equations

In future examples, I shall want to name sets of equations. I do so with let:

\[
\begin{align*}
  \text{let accounts} &= \{ A2 = 2000, A3 = 2001, B2 = 1492, B3 = 1560, \\
    &\quad C2 = 971, C3 = 1803, D2 = C2-B2, D3 = C3-B3 \}. \\
  \text{let tax} &= \{ E2 = D2\times0.33, E3 = D3\times0.33 \}. \\
  \text{accounts □ tax}
\end{align*}
\]

5.4 The @ operator: extracting subsets from a set of equations

Suppose we want to insert a column in the middle of a spreadsheet. This needs us to split parts out of the spreadsheet. I do so with @, which takes a set of equations and a cell range, and returns those equations whose left-hand sides lie within the range. Thus:

\[
\begin{align*}
  &\{ A2 = 2000, A3 = 2001, B2 = 1492, B3 = 1560, \\
    &\quad C2 = 971, C3 = 1803, D2 = C2-B2, D3 = C3-B3 \} @ A1:D2 = \\
  &\{ A2 = 2000, B2 = 1492, C2 = 971, D2 = C2-B2 \}
\end{align*}
\]
We can then insert a column this way:

\[
\text{accounts} @ A:C
\]
\[
( \text{accounts} @ D:D ) @ (1,0)
\]
\[
\{ D2 = E2 \times 0.33, \ D3 = E3 \times 0.33 \}
\]

5.5 Building one spreadsheet from two

With @, we can create a chimera. Suppose we have another spreadsheet accounts2, which has one column of expenses data in B. And suppose we want to use this in accounts. We can glue the pieces together like this:

\[
\text{let expenses2} = \text{accounts2} @ B.
\]
\[
\text{accounts} @ A:A @ \text{expenses2} @ \text{accounts} @ C:D
\]

See how naturally we now dissect components out of spreadsheets and recombine them.

5.6 Using @ to combine non-contiguous ranges

The @ operator makes sense with non-contiguous ranges:

\[
\text{let expenses2} = \text{accounts2} @ B.
\]
\[
\text{accounts} @ (A:A,C:D) @ \text{expenses2}
\]

or even

\[
\text{accounts} @ (A:A,C:D) @ \text{accounts2} @ B
\]

5.7 The mapping operator: changing layouts

When reusing part of a spreadsheet, we may need to change its layout to fit with another spreadsheet. We also need to change layouts when compiling and decompiling. For this, I introduce the mapping operator: it takes a set of equations on its left, and a source and a target cell range on its right, and rewrites the equations so that elements in the source are replaced by corresponding elements in the target.

Suppose we have a spreadsheet accounts3. Like accounts and accounts2, this holds expenses data, but running horizontally from A1 to A2. To use this in accounts, we need to rotate this row onto B2:B3. Then we can do so with mapping:

\[
\text{let expenses3} = \text{accounts3} @ A1:A2.
\]
\[
\text{let expenses3}_\text{rotated} = \text{expenses3} \text{ mapping A1:A2 to B2:B3}.
\]
\[
\text{accounts} @ A:A @ \text{expenses3}_\text{rotated} @ \text{accounts} @ B:D
\]

5.8 Arrays in equations

It's essential we can combine intelligible specifications like that of Section 4.2 with spreadsheets. We may want to decompile existing spreadsheets into such specifications and use those as the master code. So we must be able to use arrays in equations. We do so very naturally:
let accounts_S = { Year[2000] = 2000, Year[2001] = 2001,
                  Expenses[2000] = 1492, Expenses[2001] = 1560,

Similarly, we could write the tax formulae of Section 5.3 as:

let tax_S = { Tax[2000] = Profit[2000]×0.33,
              Tax[2001] = Profit[2001]×0.33 }.

5.9 Worksheets as arrays

It turned out convenient for us treat worksheets as arrays too. Thus the accounts
spreadsheet is actually as shown below, the cell notation being syntactic sugar:

let accounts = { Sheet1[1,2] = 2000, Sheet1[1,3] = 2001,
                Sheet1[2,2] = 1492, Sheet1[2,3] = 1560,
                Sheet1[3,2] = 971,  Sheet1[3,3] = 1803
                Sheet1[4,2] = Sheet1[3,2]-Sheet1[2,2],

5.10 The × operator: replicating sets of equations

This replicates an object along a new dimension:

{ y[1,2000] = 1, y[1,2001] = 1 }

This is a good way to build tables with repeated parts, such as a spreadsheet holding one
copy of a chain-store's accounting proforma for every branch of the store.

5.11 The / operator: dividing or quotienting sets of equations

The operator / is the converse of multiplication. It takes the quotient of an object,
projecting it through its final dimension:

{ y[1] = 1 }

For this to be possible, all the formulae that project onto the same result formula must be
equivalent. Variants of this operator will probably be useful in concisely specifying
searches for repeated structure when decompiling spreadsheets.

5.12 The contents_of function: code reuse and spreadsheet libraries

At last, I shall show how to reuse code from Excel files and from files holding sets of
equations. To do so, I hall introduce the contents_of function. This takes a string as
argument, assuming it to name a file. If the filename ends in .xls, we have an Excel file;
if in .exc, we assume it to hold equations written as in this paper.

This gives us libraries and code reuse. Look again at the example from Section 5.3:
let accounts = { A2 = 2000, A3 = 2001, B2 = 1492, B3 = 1560,
C2 = 971, C3 = 1803, D2 = C2-B2, D3 = C3-B3 }.
let tax = { E2 = D2×0.33, E3 = D3×0.33 }.

Now rewrite it as

let accounts = contents_of( "accounts.xls" ).
let tax = contents_of( "tax.xls" ).

In general, we can call contents_of anywhere a set of equations is needed. These equations could have been written by hand or pulled from existing spreadsheets. In either case, they could either be "raw" spreadsheet equations using cell names, or equations to be compiled, written using meaningful arrays and names. We have achieved code reuse and modularity.

6 EXCELSIOR

Excelsior is an executable equivalent of the last section: the nearest a keyboard can come to the notation. It is functional, meaning it treats every command as an expression to be evaluated. For instance:

let accounts = { A2 = 2000, A3 = 2001, B2 = 1492, B3 = 1560,
C2 = 971, C3 = 1803, D2 = C2-B2, D3 = C3-B3 }.

Excelsior data types include numbers, sets, lists, finite mappings, vectors (useful for representing offsets in spreadsheets), matrices, cell addresses within worksheets, cell ranges, worksheets, formulae or expressions, and equations, as well as (since Excelsior is built on Prolog), arbitrary Prolog terms - and spreadsheets. Typing is dynamic.

6.1 Design principles

The reasons for Excelsior's design are explained in Section 3: Excelsior provides modularity by implementing the operators of Section 5. It imitates "scripting languages" such as Perl and Python, by, for example, being a functional language, having dynamic typing, and being concise. It is interactive, and supports trial-and-error assignment of results generated during structure discovery, in the same way that Mathematica and other systems do for computer algebra - [Daly et. al.] compares approaches.

6.2 What Excelsior contains

To meet the page limit in these proceedings, I give a short general account rather than, as originally submitted, a detailed specification with examples. Excelsior includes:

Primitives: The operators of Section 5, and the notation for sets of equations.

Variables: These are defined and assigned with the let keyword, as in Section 5.
**Getting formulae:** when editing and patching spreadsheets, we often need to get individual formulae. There is a function `lookup` for this.

**Changing how formulae are interpreted:** When Excelsior reads a spreadsheet, it parses the formulae, converting to expression trees and fixing up relative cell references. Though often the most useful representation, we can ask for others. `Raw` is the formula as a string, for example `R[-33]C+1`. Strings are faster to compare than expression trees, so this is useful when we only need to compare two formulae. `Relative` is like the default representation: the formula as an expression tree, but without fixing up relative references. This is useful in structure discovery, because Excel preserves relative cell references when the user copies a formula from one cell to another. And `fully substituted` is the formula with names replaced by the corresponding item from Excel's named-range list, and with array formulae fixed up.

**Getting style information:** There are functions similar `lookup` for getting stuff such as cell styles and colours, column widths, and named ranges.

**Listing spreadsheets:** The function `show` lists a spreadsheet's equations to terminal or file in human-readable format. This is a handy way to see a spreadsheet's calculations, and to get them printed.

**Loading and saving files:** The function `load` is the Excelsior equivalent of Section 5's `contents_of`. There is also a function for saving spreadsheets to file.

**Running Excel:** The function `excel` runs Excel on a spreadsheet.

**Editing formulae:** The function `replace` acts like a text editor's global search-and-replace, on individual formulae. As an example, I built a small algebraic simplifier which replaces a spreadsheet’s formulae by simplified equivalents. I also used it in the accounts-spreadsheet patching trial of Section 9.2, to replace all references to original inputs by the corrected equivalents.

**Spreadsheet grammar rules:** These are described in [Paine, 2004(a)].

**Spreadsheet emulation:** The function `evaluate` is a simple spreadsheet engine. It is far from a complete implementation of the Excel functions, being limited to the basic arithmetic operators and mathematical functions, but it was still useful in running some economic-simulation spreadsheets in a Web-based distance learning system. For a more complete evaluator, I am considering Gnumeric [Gnumeric].

### 7 PARAMETERISATION, GENERALISATION, END-USERS, AND GRAPHICAL USER INTERFACES

So far, I have written as though each module will do the same thing every time it is used. However, we shall often want to change, for instance, the cells from which modules take their inputs, and the size of the tables they work on. So we must be able to give them parameters – to tell a module which cells are its inputs, or how big a table is. This can be done by defining Excelsior functions with such arguments.

I have also written as though Excel users will build spreadsheets by typing Excelsior commands. This is not so. Excelsior can be used directly by those with enough
programming experience. I want, however, to provide something for those who know only Excel. To do this, I am working on automatically decomposing existing spreadsheets into modules, then generalising these modules so that they can be parameterised as above. With these, users will be able to build spreadsheets and have them automatically converted into reusable modules. This will need a graphical user interface to display newly proposed modules and to splice them into existing spreadsheets. I shall report this work in a future paper. The source of Excel not being available, Gnumeric [Gnumeric] may be a useful vehicle for experiments.

8 EXCELSIOR AND PROLOG

Excelsior is implemented in Prolog [Paine, 2004(b)]. For the reasons stated in Section 6.1, I did not make it resemble Prolog. However, some spreadsheet engineers will be willing to learn Prolog: it is, for example, extremely useful for representing structural relations between spreadsheet modules. Therefore, Excelsior enables its user to drop into the Prolog top-level interpreter, and to call Prolog predicates as Excelsior functions.

9 APPLICATIONS

Excelsior is new, but there has been time to try four applications. That of Section 9.4 was requested for evaluation by a large US investment bank and, in part, involves checking a spreadsheet for allocating loans of up to 5 billion dollars. The other three are small trials to see how well Excelsior fitted certain tasks. The first, reverse-engineering a cellular automaton, shows how a huge spreadsheet can be compressed into a tiny listing, which can then be parameterised and compiled to generate variants of this spreadsheet.

9.1 Reverse-engineering and simplifying a cellular automaton

This trial used the small-worlds cellular automaton of [Complex Analysis]; [Hand, 2005] explains how such spreadsheets work. Because it lays out 26 successive generations of the cellular automaton down one worksheet, the small-worlds spreadsheet holds a vast amount of repetitive structure, both within each generation's grid, and from one grid to another. Using Excelsior functions for detecting and grouping identical formulae, I could list the spreadsheet very compactly, reducing a 1.8 MB Excel file to a 14K plain text listing containing 17 equations. I was then able to change the state-change formulae in this listing and feed it through the Excelsior compiler, generating a spreadsheet simulating a new cellular automaton. To generate such a new cellular automaton directly in Excel, I would have had to edit about 800 formulae in the original spreadsheet.

As an example, Excelsior listed one batch of repeated formulae as:

\[
\text{Sheet1}\{\text{(1)}\} = \text{Sheet1} + 1
\]

This says that the cells in rows 37, 70, ... 829 of column 1 contain the relative formula \(R[-33]C+1\).

9.2 Patching end-of-year accounts

A trial suggested by spreadsheet expert Duncan Williamson used Excelsior to edit accounts spreadsheets. These often fail to balance at the end of the accounting year, because some credits or debits are wrong. Accountants fix this by patching with
corrections such as "The sales department must be noted to have spent an extra £200". To do these edits directly in Excel, we would have to insert extra rows containing the correction, then update all cell references so they refer to these corrected values rather than the originals. In contrast, it was easy to code the edits in Excelsior, and also to reformat the spreadsheet to the cascaded layout Williamson recommends [Williamson].

9.3 Converting spreadsheets to other programming languages

I have experimented with Excelsior to dump spreadsheet-based simulations as equations for reimplementation in other programming languages. These include Java, Fortran, and Lumina's Analytica modelling package.

9.4 Comparing and stylechecking an investment bank's credit application and other spreadsheets

A large and well-known US investment bank is evaluating Excelsior for stylechecking and security. One stylechecking program written in Excelsior checks spreadsheets follow the bank's guidelines that only one copy of any formula occurs on each worksheet. Another application concerns a spreadsheet, copies of which businesses fill in when applying for a loan. To check the applicants haven't tampered with formulae in their copy, I have written an Excelsior program for comparing copies against the original. This application is crucial, as the spreadsheet is used for loans of up to 5 billion dollars.

10 ACKNOWLEDGEMENTS

To Andreas, Jack and Kostas, at the Excelsior café, Cowley Road. Something that's excelsior does more then excel.

11 REFERENCES


16 A Revised Classification of Spreadsheet Errors

Kamalasen Rajalingham,
University of Westminster, 115 New Cavendish Street, London W1W 6UW,
K.Rajalingham@wmin.ac.uk

ABSTRACT

This paper describes an improved and revised framework for the systematic classification of spreadsheet errors. In past publications, the derivation of the taxonomic scheme and the justification for the proposed approach were not discussed in adequate detail. The new revised classification addresses these limitations and presents clearer guidance on the classification of specific types of spreadsheet errors. Like the previous classification schemes, this revised taxonomy of errors is also aimed at facilitating a more thorough analysis and understanding of the different types of spreadsheet errors. It is more comprehensive than any presented or published before. Every class or category of errors is explained and supported by appropriate examples. The ability to place an error within a certain class in the taxonomy should enable us to understand other similar errors and devise a strategy to prevent their occurrence.

1. INTRODUCTION

As mentioned in previous publications [Rajalingham, 1999, 1999a, 2002; Chadwick, 1999], the phenomenon and magnitude of spreadsheet errors can be viewed or investigated from three distinct perspectives: frequency of spreadsheet errors, real-life consequences of these errors, and the occurrence of specific types of spreadsheet errors. This paper is mainly related to the third perspective, concentrating on the examination and classification of specific errors. This area has gained very little attention in the past, resulting in a lack of understanding of the nature and causes of spreadsheet errors. An analysis of specific types of errors should precede the development of strategies and solutions to deal with the problem effectively. This involves the classification of these errors.

The previous version of the taxonomy of spreadsheet errors was presented by Rajalingham et al [Rajalingham, 2000a]. This paper presents a more comprehensive classification of spreadsheet errors than ever presented or published before, following a meticulous analysis of the nature and characteristics of specific types of spreadsheet errors from a wide variety of sources. This classification is confined to only user-generated spreadsheet errors, as opposed to the occurrence of errors produced by the spreadsheet software, which is beyond the scope of the current research. The classification is based on a rational taxonomic scheme and is supported by a selection of generic and specific examples. Earlier versions of the taxonomy have been published [Rajalingham, 1998, 1999, 1999a, 2000].
2. THE CONCEPT OF TAXONOMY OR CLASSIFICATION

The concept of taxonomy or classification has been discussed in some detail by Rajalingham et al. [Rajalingham, 2000a], mainly quoting Britannica.com [Britannica.com, 1999, 2000]. According to these sources, taxonomy refers to the science of classification, which is usually applied to the classification of living and extinct organisms. However, there is no special theory behind modern taxonomic methods [Britannica.com, 1999, 2000].

According to Britannica.com [Britannica.com, 1999, 2000], in biology, taxonomy refers to the establishment of a hierarchical system of categories on the basis of presumed natural relationships among organisms. The same source further states that the goal of classifying is to place an organism into an existing group or to create a new group for it. Rajalingham et al. [Rajalingham, 2000a] adopted these definitions and extended the concept of taxonomy to the classification of spreadsheet errors. The spreadsheet error taxonomy can be defined as a hierarchical system of classes of spreadsheet errors on the basis of common characteristics and relationships.

3. RATIONALE FOR THE CLASSIFICATION OF SPREADSHEET ERRORS

There are various reasons for developing a classification of spreadsheet errors. It is a methodical approach to problem analysis. The analysis of the different types of errors based on this approach is likely to improve comprehensive testing of a spreadsheet development methodology. The classification of spreadsheet errors also enables us to gain a deeper understanding of the characteristics of an error as well as the nature of its occurrence. A comparison can also be made with other related errors belonging to the same class.

An insight into the characteristics and nature of an error is extremely important, in order to prevent the occurrence of the error or develop a method for detecting its presence. The identification of similar characteristics and properties between errors, may enable the development of similar approaches to deal with spreadsheet errors within the same taxonomic group. Knowledge of the characteristics of an error can also help in evaluating its potential impact and frequency, probably shared by other errors in the same category.

4. DERIVATION OF THE TAXONOMIC SCHEME

As there is no special theory behind modern taxonomic methods, methods of classification employed in other fields can be used to guide the process of classifying spreadsheet errors. Based on the principles of classification adopted in zoology and botany [Britannica.com, 1999, 2000], spreadsheet errors can be classified using a similar taxonomic scheme, consisting of the following steps:

- A specific type and example of a spreadsheet error is obtained.
- The error is compared with the known range of variation of spreadsheet errors.
- The error is correctly identified if it has been described, or a description showing similarities to and differences from known categories, is prepared. If the error is of a new type, it is assigned to a new category or class.
- The best position for the error is determined in the existing classification. This may also involve determining what revision the classification requires as a consequence of the new discovery.
Available evidence is used to further suggest and describe the nature of the error, its possible causes and other characteristics.

Based on Britannica.com [Britannica.com, 1999, 2000], it is clear that the process of spreadsheet error classification requires a recognised system of ranks, rules and a verification procedure. An investigation of a taxonomic method that addresses these requirements revealed that there are two possible approaches to structuring the ranks within a taxonomy, a binary approach or a bushy approach. Both methods are based on a top-down approach that produces a hierarchical taxonomy, by studying the nature and characteristics of errors.

The bushy approach was initially adopted and assessed. A category at any level or rank can be divided into two or more classes. An example of an earlier version of the proposed taxonomy using the bushy approach is shown in Figure 1 [Rajalingham, 1999, 1999a, 2000; Chadwick, 1999]. The bushy taxonomic structure shown in Figure 1 was found to have certain limitations. It was difficult to navigate down the taxonomic tree to assign a specific error to a class. It was also possible to place certain errors in two or more different classes, potentially resulting in an ambiguous interpretation of the errors.

![Figure 1: Taxonomy Using a Bushy Approach](image)

In order to address these limitations, the alternative binary approach was considered. At each stage of the taxonomy, the binary approach uses dichotomies or division into two mutually exclusive (non-overlapping) groups, to classify the errors. This minimises the possibility of positioning the same type of error in different classes/sub-classes and causing an overlap. This feature of the binary approach makes it a far more straightforward way of assigning a specific error to a taxonomic class. A simple IF-THEN-ELSE rule or constraint can be used to navigate down the taxonomy tree and position errors in appropriate classes. In order to reduce ambiguity, for each dichotomy, only a single
factor, representing a distinct aspect of the error, is considered. To this end, the following aspects of a particular type of spreadsheet error are analysed:

- Manifestation of the error
- Cause of the error
- The role of the person responsible for the error
- The cognitive state of the person responsible for the error
- The stage of the spreadsheet building life cycle where the error occurs
- The relevant view of the spreadsheet model system

In view of the advantages of the binary method compared to the bushy method, the binary approach has been adopted as the basis of a rational taxonomic scheme for classifying spreadsheet errors.

5. THE CLASSIFICATION OF SPREADSHEET ERRORS

The term error used in this paper has a broader definition encompassing both actual errors and potential errors. The errors include flaws, slips and mistakes. In the process of classifying certain specific errors, assumptions are made about the precise cause of the errors, where this is not clearly described by the source. Otherwise, it would be possible to assign the same error to several different categories.

Based on the new revised classification of user-generated spreadsheet errors, at the highest level, spreadsheet errors can be divided into two non-overlapping categories of quantitative and qualitative errors. The classification factor used at this stage is the manifestation of the error. Panko and Halverson [Panko, 1996] have also broadly split spreadsheet errors into quantitative or qualitative errors.

For all user-generated spreadsheet errors:

IF numerical error causing incorrect bottom-line value
THEN quantitative error
ELSE NOT quantitative error (i.e. qualitative error)

5.1 Quantitative Errors

Quantitative errors are numerical errors that lead to incorrect bottom-line values [Panko, 1996]. They simply produce wrong data in the spreadsheet model. Based on an analysis of the cause of the error, a dichotomy of accidental and reasoning errors can be used to capture the different types of quantitative errors. Any error or flaw, which is quantitative and not accidental, must have been produced as a result of a mistake in reasoning.

For all quantitative errors:

IF error is caused by negligence or carelessness
THEN accidental error
ELSE NOT accidental error (i.e. reasoning error)

The dimension of fraud is not taken into account when developing the classification framework for quantitative errors, as any error can be deliberately produced with fraudulent or malicious intent and disguised as an accidental or reasoning error.
1. Accidental Errors

*Accidental errors* are mistakes and slips caused by negligence, such as typographical or pointing errors. Though quite frequently occurring, they have a high chance of being spotted and corrected immediately. Based on the user role responsible for the error, an accidental error can either be a *structural error* or a *data input error*.

For all *accidental* errors:
- IF error is caused by the model developer
- THEN *structural error*
- ELSE *NOT structural error* (i.e. *data input*)

(a) Structural Errors

*Structural errors* are errors produced by the developer of the spreadsheet model. These errors are produced when creating or altering the structural or programmed component (formula network) of the spreadsheet model. Therefore, these errors can be further segregated into two categories, namely, *insertion* and *update* errors.

For all *structural* errors:
- IF error is produced when creating the structural aspects of the spreadsheet model
- THEN *insertion* error
- ELSE *NOT insertion* error (i.e. *update* error)

(i) Insertion Errors

These errors occur while the developer is creating the structures of the spreadsheet model. The model would be prone to accidental errors such as typographical errors, pointing errors, duplication and omissions.

*Example 1: Omissions*

Omissions are important factors or variables that are left out of a spreadsheet model [Cragg, 1993]. According to Panko and Halverson [Panko, 1996], research had shown that omission errors were dangerous due to the low detection rates.

*Example 2: Pointing Errors*

Pointing errors refer to errors caused by references being made to wrong cells or cells in the wrong location. The model developer types the wrong cell coordinates in composing the formula [Brown, 1987]. As a result, the formulae themselves produce incorrect results and may even refer to blank cells or non-numeric cells.

(ii) Update Errors

These errors occur while the developer is altering the structural or programmed component (formula network) of the spreadsheet model. The model at this stage would be prone to accidental errors such as typing errors, overwriting and deletion.
For all structural update errors:
  IF If error is produced as a result of incorrectly changing the structural or programmed component of the model
  THEN modification error
  ELSE NOT modification error (i.e. deletion error)

Modification Errors

These errors occur as a consequence of incorrectly or inaccurately modifying the structural or programmed component (formulae) of the spreadsheet model. The modification of spreadsheets is more prone to errors compared to the original creation of spreadsheets [Brown, 1987].

Example 1: Formulae Overwritten with Data

Data is incorrectly entered into a cell previously containing a formula, overwriting the formula and invalidating the model [Cragg, 1993]. Stang [Stang, 1987] and Hayen [Hayen, 1989] also described this error. A simple solution to the problem is to use cell protection.

Example 2: Formula Overwritten with an Incorrect Formula

Similar to the previous example. However, the correct formula is accidentally replaced with an erroneous formula.

Deletion Errors

These update errors, on the other hand, are produced as a result of deleting or erasing existing elements of the structural or programmed component of the spreadsheet model.

Example 1: Deletion of a Formula

A correct formula required by the spreadsheet model is accidentally erased. The main cause of this error is the failure to protect cells containing formulae.

(b) Data Input Errors

Data input errors are errors made by end-users who merely manipulate the spreadsheet model. They are caused by erroneous entry of data required by the model. These errors can occur while either inserting new data or amending/updating existing data.

For all data input errors:
  IF error occurs when entering new data into the spreadsheet model
  THEN insertion error
  ELSE NOT insertion error (i.e. update error)

(i) Insertion Errors

These errors are produced while entering new data into the model. Typically these would take the form of typographical errors or omissions committed by the data entry users.
**Example 1: Erroneous Data Input**

Invalid or incorrect data is easily entered into the spreadsheet model because there are no data checks on entry. Sometimes, the right data is put in the wrong cell. Wrong data can occur either due to a data entry error or incorrect data held by the data source [Hayen, 1989]. Freeman [Freeman, 1996] proposes the use of limit controls (tolerable ranges) to deal with these errors.

**Example 2: Omissions**

It is not uncommon for data entry operators to accidentally leave out certain inputs to the model.

(ii) Update Errors

These errors are produced as a result of incorrectly updating existing data in the model. Update operations (apart from insertion) must either be modification (or overwriting) or deletion.

For all update errors:

IF error occurs as a result of overwriting existing data
THEN *modification* error
ELSE NOT modification error (i.e. *deletion* error)

Modification Errors

These errors are produced as a result of changing existing data in the model. Typically these would be typographical or overwriting errors committed during data entry.

**Example 1: Overwriting of Data**

A correct piece of data entered is overwritten with an incorrect input. This might be caused by an update being applied in the wrong location.

Deletion Errors

These errors, on the other hand, occur as a result of deleting or erasing previously entered data from the model. These errors are also caused by users responsible for data entry.

**Example 1: Erasure of Data**

A correct piece of input required by the model is simply deleted inadvertently. This is usually done during data entry or update.
2. Reasoning Errors

Reasoning errors are mistakes in reasoning and therefore not accidental. They are produced as a result of a lack of knowledge required to comprehend, analyse and accurately model the business function or problem in the form of a spreadsheet model. Reasoning errors can be split into two distinct categories based on an analysis of the precise cause of the errors, which in this case also involves a study of the cognisance of the model developer. The two mutually exclusive classes of reasoning errors are domain knowledge errors and implementation errors.

Any reasoning error which is not produced owing to inadequate comprehension of the underlying problem or function to be modelled on the spreadsheet, could only possibly have been caused by an incorrect implementation of the problem or function using the spreadsheet package. Implementation errors are far more common than domain knowledge errors, though domain knowledge errors are generally more serious.

For all reasoning errors:

IF error occurs owing to a lack of understanding of the underlying problem or function to be modelled
THEN domain knowledge error
ELSE NOT domain knowledge error (i.e. implementation error)

(a) Domain Knowledge Errors

Domain knowledge errors are specifically caused by inadequate awareness or knowledge required to identify, analyse and understand the business function or problem underlying the spreadsheet model. This knowledge is essential for modelling the problem and designing the corresponding conceptual or logical data model.

This category of errors consists of two distinct classes, namely real-world knowledge and mathematical representation based errors. Any reasoning domain-knowledge error which occurs despite selection of the right algorithm must have been caused by a lack of understanding of how the algorithm is to be mathematically represented.

For all domain-knowledge errors:

IF error caused as a consequence of a lack of knowledge on the underlying algorithm of a calculation or function
THEN real-world knowledge error
ELSE NOT real-world knowledge error (i.e. mathematical representation error)

(i) Real-world Knowledge Errors

These errors involve creating a formula by selecting the wrong algorithm. Users may select an inappropriate template for a particular analysis or decision task, due to a lack of accounting knowledge or intellectual modelling logic.

Example 1: Exclusion of Factors from Formulae

A fairly common error in this category is the exclusion of important factors in a calculation. For instance, bad debt provision is excluded in an accounting calculation.
Example 2: Absence of Distinction Between Leap and Non-leap Years

This is a simple example of a real-world knowledge error whereby to calculate the daily figures for a particular leap year, the calculations divide by 365 instead of 366.

(ii) Mathematical Representation Errors

These errors involve constructing the wrong formula despite having selected the right algorithm. This is due to a lack of knowledge on how to represent a mathematically correct and accurate formula based on the correctly chosen algorithm.

Example 1: The PERCENTAGE Error

This error occurs when the formula to calculate percentage is incorrectly written, due to a lack of knowledge of how to calculate a percentage or BODMAS (Brackets, Of, Division, Multiplication, Addition, Subtraction), which identifies precedence in calculations.

Example 2: Incorrect Representation of an OVERALL AVERAGE Function

Based on Figure 2, the correct formula in F9 is =E9/B9 but the formula =AVERAGE(F5:F8) is entered instead [Chadwick, 1997, 1997a]. Although the model developer knew that an overall average was to be calculated, they incorrectly assumed that the sum of averages would give the overall average.

<table>
<thead>
<tr>
<th>A</th>
<th>B</th>
<th>C</th>
<th>D</th>
<th>E</th>
<th>F</th>
</tr>
</thead>
<tbody>
<tr>
<td>Lazy Days</td>
<td>Staff Numbers</td>
<td>Basic Wages £</td>
<td>Overtime Wages £</td>
<td>Total Wages £</td>
<td>Average Wages £</td>
</tr>
<tr>
<td>---</td>
<td>---</td>
<td>---</td>
<td>---</td>
<td>---</td>
<td>---</td>
</tr>
<tr>
<td>1</td>
<td>Costs 1995-1996</td>
<td>---</td>
<td>---</td>
<td>---</td>
<td>---</td>
</tr>
<tr>
<td>2</td>
<td>---</td>
<td>---</td>
<td>---</td>
<td>---</td>
<td>---</td>
</tr>
<tr>
<td>3</td>
<td>---</td>
<td>---</td>
<td>---</td>
<td>---</td>
<td>---</td>
</tr>
<tr>
<td>4</td>
<td>---</td>
<td>---</td>
<td>---</td>
<td>---</td>
<td>---</td>
</tr>
<tr>
<td>5</td>
<td>Managers</td>
<td>1</td>
<td>17700</td>
<td>0</td>
<td>---</td>
</tr>
<tr>
<td>6</td>
<td>Grade 1</td>
<td>3</td>
<td>45540</td>
<td>1400</td>
<td>---</td>
</tr>
<tr>
<td>7</td>
<td>Grade 2</td>
<td>9</td>
<td>122340</td>
<td>2000</td>
<td>---</td>
</tr>
<tr>
<td>8</td>
<td>Grade 3</td>
<td>12</td>
<td>102350</td>
<td>0</td>
<td>---</td>
</tr>
<tr>
<td>9</td>
<td>Grand Totals</td>
<td>25</td>
<td>287930</td>
<td>3400</td>
<td>---</td>
</tr>
</tbody>
</table>

Figure 2: Example to Illustrate Overall Average Error

(b) Implementation Errors

Implementation errors are produced due to a lack of knowledge or understanding of the full use of the functions and capabilities of the particular spreadsheet package in use, with an understanding of the spreadsheet principles, concepts, constructs, reserved words and syntax. Implementation errors consist of logic and syntax errors.

For all implementation errors:

IF error is caused by a lack of comprehension of the features and functions of the spreadsheet package/language

THEN logic error

ELSE NOT logic error (i.e. syntax error)
(i) Logic Errors

Logic errors are errors caused by a lack of understanding of the functions and capabilities of a specific spreadsheet package, which enable the accurate representation of a solution.

**Example 1: RELATIVE and ABSOLUTE Copy Problem**

The relative copy causes cell references in a copied formula to alter row and column references relative to the original cell copied [Chadwick, 1997]. The error is also caused by copying a formula hidden underneath a cell value, thinking that it is the value that is being copied [Brown, 1987].

**Example 2: Value Not Included in the Total**

This error has been pointed out by several authors [Ayalew, 2000; Butler, 1997; Stang, 1987; Ditlea, 1987]. The modeller writes a formula to find a range total in cell B10. The formula is =SUM(B1:B9), and data are entered in cells B1 to B9. A row is then inserted below cell B9 and a new value entered in B10. This cell is beyond the range of the formula (which has now been shifted to B11) and therefore not included in the addition.

**Example 3: Rounding Error**

Rounding can and should always be controlled. The best approach is to perform all operations on rounded numbers, and not with “hidden” or formatted values. Based on Figure 3 [Batson, 1991], it can be seen that the “formatted” column does not add up, and therefore affects the credibility of the model.

<table>
<thead>
<tr>
<th></th>
<th>Actual</th>
<th>Formatted</th>
<th>Rounded</th>
</tr>
</thead>
<tbody>
<tr>
<td>A1</td>
<td>1.128431</td>
<td>1.13</td>
<td>1.13</td>
</tr>
<tr>
<td>A2</td>
<td>2.35625</td>
<td>2.36</td>
<td>2.36</td>
</tr>
<tr>
<td>A3</td>
<td>1.827994</td>
<td>1.83</td>
<td>1.83</td>
</tr>
<tr>
<td>=SUM(A1:A3)</td>
<td>5.312675</td>
<td>5.31</td>
<td>5.32</td>
</tr>
</tbody>
</table>

**Figure 3: Rounding Error**

(ii) Syntax Errors

Syntax errors are errors caused by a lack of precise understanding of the constructs, reserved words and syntax of a specific spreadsheet package, used to write functions and formulae.

**Example 1: A Keyword Within a Formula is Misspelled**

A keyword within a formula is misspelled causing an error, e.g. =AVG(...) instead of =AVERAGE(...). This error can be easily detected.
5.2 Qualitative Errors

Qualitative errors are errors that do not immediately produce incorrect numeric values but degrade the quality of the model. The model also becomes more prone to misinterpretation, and difficult to update/maintain. Qualitative errors can be generally divided into two different types, namely, temporal errors and structural errors.

This dichotomy is obtained mainly based on an analysis of the three views of an information system: data, processing and behaviour. Within the context of spreadsheet models, the processing view of a model is the network of formulae used to perform calculations, while the data view represents the various input data required for the calculations. The behavioural or temporal view represents the effects of time and real world events on the spreadsheet model. A qualitative error which is not temporal can be considered a structural error. The structural aspect of the model represents the formula network and data.

For all qualitative errors:

IF error is caused by an elapse of time, which invalidates data
THEN temporal error
ELSE NOT temporal error (i.e. structural error)

1. Temporal Errors

Temporal errors are qualitative errors which invalidate data (and possibly formulae) with the passage of time. As a result, the model ceases to be reliable. Typically these errors are caused by failure or delays in updating the spreadsheet model to reflect current circumstances.

Example 1: Qualitative Error Caused by the Referencing of Non-current Data

This error is produced as a result of referencing a piece of data that has become invalid/inaccurate due to time lapse.

Example 2: A Previous Erroneous Model is Used

There may be different versions of a model, where each version may have been revised differently by some user [Stang, 1987]. It is possible that the most current version has been fully debugged, but a previous version with flaws is used to make important decisions.
2. Structural Errors/Flaws

Structural errors in this context can be defined as non-temporal qualitative errors or flaws produced as a result of poor design or layout of model structures and data. Based on the physical manifestation of these errors, they can be divided into two categories: visible errors and hidden errors.

For all structural errors:

IF error is a structural flaw which is visible at the surface level of the model
THEN visible error
ELSE NOT visible error (i.e. hidden error)

(a) Visible Errors/Flaws

Visible qualitative structural errors are structural flaws which are visible at the surface level of the model. The detection of these errors does not require any examination of the formula level. These errors normally take the form of semantic errors which make the models more prone to misreading or misinterpretation.

Example 1: Formatting Error

A common qualitative error is where the cell format is specified as general on the spreadsheet. Consequently, the figures have varying decimal places and make it difficult to identify a number that is incorrect. This is shown in Figure 4. The value in cell G10 is greater than the value in cell G9. However, at a quick glance, the value in cell G9 (102350.25) may seem to be greater than the value in cell G10 (291331.3) due to the inconsistent use of decimal places [Chadwick, 1997].

(b) Hidden Errors/Flaws

Hidden qualitative structural errors, on the contrary, are structural flaws which are not visible at the surface level of the model and therefore require examination of the formula level. These errors normally take the form of complicated, confusing or inappropriate construction of formulae. Such flaws can make the model difficult to maintain and prone to inconsistencies or update anomalies.

Example 1: Hard-coding

A fixed value is used when a variable (cell reference) should be used instead. In other words, the cell contains a hard-coded input instead of a formula. For instance, net * 17.5% instead of net * a named variable or range for VAT rate [Butler, 2000].
Example 2: Complexity of Formulae

Stang [Stang, 1987] states that any equation longer than 80 characters uses logic that is difficult to follow.

6. CONCLUSION

In spite of an increasing awareness of the consequences of user-generated spreadsheet errors, there has been a lack of research and analysis of specific types of these errors. In order to effectively deal with the problem, a thorough examination and classification of specific types of spreadsheet errors is essential.

This paper has described a more comprehensive taxonomy of user-generated spreadsheet errors than ever presented or published before, based on a rational taxonomic scheme. The systematic classification of spreadsheet errors enables us to gain a far better understanding of the different types of errors. This facilitates the development of tools, techniques and methods to prevent their occurrence or improve the detection of existing errors.

7. REFERENCES


Comparison of Spreadsheets with other development tools (limitations, solutions, workarounds and alternatives)

Simon Murphy,
Codematic Limited,
Kinkry Hill House
Roadhead
Carlisle
UK
simon.murphy@codematic.net

ABSTRACT

The spreadsheet paradigm has some unique risks and challenges that are not present in more traditional development technologies. Many of the recent advances in other branches of software development have bypassed spreadsheets and spreadsheet developers.

This paper compares spreadsheets and spreadsheet development to more traditional platforms such as databases and procedural languages.

It also considers the fundamental danger introduced in the transition from paper spreadsheets to electronic.

Suggestions are made to manage the risks and work around the limitations.

1. INTRODUCTION

“400 Million users cant be wrong!” (Microsoft, 2005)

Spreadsheet usage is almost universal (or endemic – depending on your view point). If cashflow is the lifeblood of business, spreadsheets are the language. They enjoy rather more widespread use than the paper original ever did. They are currently used for analysing, modelling, reporting and forecasting billions and billions of pounds worth of business transactions daily. There is no evidence of widespread business collapses due to spreadsheet errors (but the odd bankruptcy has been known).

Spreadsheets are being used for more and more ambitious projects, many are well beyond that envisaged by the original creators in the 70’s. Are we beyond the limit? In the 70’s ‘Garbage In, Garbage Out’ was the standard. In 2005 that is just not acceptable ‘Garbage In, Error message out’ or possibly even better ‘No Garbage in’ is the modern standard. (McConnell, 2004)

This paper compares spreadsheets to other development tools, looks at some of the problems associated with spreadsheets as a development platform and suggests workarounds and solutions.
2. COMPARISONS WITH OTHER TOOLS

Select by location not value.

Paper spreadsheets have 1 mode of access, by the user, by value (you look down the title column looking for the text that describes the items you want). Electronic spreadsheets have 2 modes of access – the same user mode, by value, and the underlying, formula based, ‘by location’ access mode. This disconnect is unique to electronic spreadsheets and is a fundamental weakness that guarantees fragile systems. This duality is a significant barrier to understanding and auditing non trivial spreadsheets.

Simple Demo:
Are the Gross Profit formulas correct?

<table>
<thead>
<tr>
<th>A</th>
<th>B</th>
<th>C</th>
<th>D</th>
<th>E</th>
<th>F</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td><strong>Simple Gross Profit calculation - enter values</strong></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>2</td>
<td>Sales</td>
<td>2003</td>
<td>2004</td>
<td>2005</td>
<td></td>
</tr>
<tr>
<td>4</td>
<td>COGS</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>5</td>
<td>Gross Profit</td>
<td>=D3-D5</td>
<td>=E3-E5</td>
<td>=F3-F5</td>
<td></td>
</tr>
</tbody>
</table>

What about these?

<table>
<thead>
<tr>
<th>A</th>
<th>B</th>
<th>C</th>
<th>D</th>
<th>E</th>
<th>F</th>
</tr>
</thead>
<tbody>
<tr>
<td>2</td>
<td>Sales</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>4</td>
<td>COGS</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>5</td>
<td>Gross Profit</td>
<td>=B4-B2</td>
<td>=D4-D2</td>
<td>=E4-E2</td>
<td></td>
</tr>
<tr>
<td>2</td>
<td>Sales</td>
<td>2003</td>
<td>2004</td>
<td>2005</td>
<td></td>
</tr>
<tr>
<td>4</td>
<td>COGS</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>5</td>
<td>Gross Profit</td>
<td>=D3-D5</td>
<td>=E3-E5</td>
<td>=F3-F5</td>
<td></td>
</tr>
</tbody>
</table>

(Yes, No, No and No)

The actual (not apparent) spatial relationship is critical to understanding and testing a spreadsheet. The appearance and layout are irrelevant at best, and often downright misleading. Row and column headers must be visible to understand the model. The connection between meaningful labels and executable logic is coincidental.

Alternative development platforms such as databases rely on a select by value approach. As in ‘SELECT * FROM PL WHERE LineItem = “Sales”’. This is much more robust. The human readable labels are used by the software.
Type Safe
Type safety is a contract that a program will not perform an operation on a variable that is not valid for that data type. Modern languages are more and more rigorously type safe. Most compilers will warn of an attempt to assign a string value to a numerical data type. Spreadsheets have no real comprehension of data types, you can put anything in any cell.

Scope
Modern programming best practice recommends minimising the visibility of variables. Block scope is preferred to routine, which is preferred to module which is preferred to global data. If an application really needs global data to function, this is a strong sign of significant design flaws (McConnell, 2004). In a spreadsheet, cells have global read visibility. Any other cell anywhere can see the value in any cell. This prevents the reliable use of information hiding and interface programming. Erwig suggests this global visibility puts spreadsheets in the same category as assembly language (Erwig, 2004).

Data separation
In N-Tier architectures there is the data tier, the business logic tier(s) and presentation tier. This separation allows each part to be optimised for its particular purpose, and minimises the effects of changes. In a spreadsheet everything is commonly lumped together, and presentation requirements often take priority over documenting complex business rules.

Security
Server based architectures are inherently more secure than desktop, and compiled binaries are difficult to modify maliciously or accidentally. Modern databases provide role based security that can be integrated with the operating system and applied at the record or field level. Worksheet protection is trivial to bypass, and often counter productive, workbook open protection is irrelevant if the user needs to open the workbook to use it.

Scalability
A program routine is written once and used many times, whereas each spreadsheet cell needs its own version of a formula. A VB program to take some numbers and add them is much more complex than a spreadsheet ‘SUM()’ formula. But the VB code to sum a thousand sets of numbers has the same complexity, a spreadsheet would have 1,000 formulas, each needing to be checked for correctness, arguably 1,000 times (or more) more complex.

Development tools
The Latest version of Visual Studio (VS 2005) assists the developer to create UML diagrams to represent the system, automatically generate database schemas and code, work with databases, write code to implement business rules, design the user interface (eg Web, windows forms, even Excel). It provides security and traceability for development resources through a source control system, it offers unit testing with automatic creation of test cases. All without leaving the development environment, all with context sensitive help and tips. Spreadsheets offer a few intrinsic tools to assist
with development and testing but the difference in scope, power and flexibility is

dramatic.
Panko suggests spreadsheet development is in a similar condition to mainstream
development in the 60’s (Panko, 1998) – and he’s right, and so are the tools.

**VBA**
Commercial Excel VBA code is generally of appalling quality, most of it breaking every
recommended best practice. The Excel/VBA link is not robust and appropriate use of
named ranges to connect code and worksheet cells is rare, string constant references are
more common.

**Ad-hoc**
Spreadsheets are a superbly powerful and flexible ad-hoc analysis and presentation tool
for a single user. The second best tool for everything (Powell, 2004). Unfortunately ad-
hoc tools lead to ad-hoc designs, ad-hoc designs are hard to test, hard to maintain, and
hard to extend. With no formal development lifecycle or migration plan, models live on
and develop beyond their initial life expectancy and scope.

**Links**
Inter-workbook links create hidden dependencies and make data consistency difficult to
assess. Links enable circular references that Excel cannot spot, unless all linked
workbooks are open at once.

Example - Analysis of link sources for 1 live, commercial workbook:

![Link Diagram]

34 linked workbooks, 20 of which were found, 14 workbooks missing so unchecked for
further links, over 100 links found. Chances of it being correct? Depends on your
definition of correct: 100%(if you mean correct enough), 0%(if you mean totally, provably correct).

3. SUGGESTIONS, SOLUTIONS AND WORKAROUNDS

General
The main advice is to be aware of the limitations of spreadsheets.

If you are a developer, then you owe it to your clients to know enough about alternative development platforms to be able to advise when a spreadsheet might not be the best choice.

If you are an end user you need to be aware of the signs your spreadsheet analysis may have outgrown its current implementation. (eg: unwieldy, difficult to modify, difficult to reconcile, incomprehensible, only usable by the original author)

If you are a manager you need to be aware of where your information comes from and how it gets to you. Teams of analysts working in Excel all day may not be creating value, they may be creating a monster that will eventually paralyse your business (the so called Spreadsheet Hell). Warning signs are long delays in answering apparently simple questions, regular errors, reports that don’t reconcile to other sources, limited team skills outside spreadsheets, aversion to working with each others models, lack of formal IT training, lack of IT department interaction.
<table>
<thead>
<tr>
<th>Issue</th>
<th>Cause</th>
<th>Impact</th>
<th>Management suggestions</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Select by location not value</td>
<td>Visual approach to modelling</td>
<td>Spreadsheet view is disconnected from the user view</td>
</tr>
<tr>
<td>2</td>
<td>Not type safe</td>
<td>Allows rapid modelling and quick changes</td>
<td>Visible representation of cell contents may be misleading</td>
</tr>
<tr>
<td>3</td>
<td>Global Scope</td>
<td>Simple reuse of previous analysis</td>
<td>Inner workings cannot be hidden to allow later changes with no side effects</td>
</tr>
<tr>
<td>4</td>
<td>Lack of Data/Logic separation</td>
<td>Reduces need for forethought and design</td>
<td>Comprehension is reduced</td>
</tr>
<tr>
<td>5</td>
<td>Lack of security</td>
<td>Primarily a single user tool</td>
<td>Intellectual property cannot be protected, spreadsheets can't be trusted once distributed</td>
</tr>
<tr>
<td></td>
<td>Poor scalability</td>
<td>lack of true data/logic separation means different data cannot be run through the same logic</td>
<td>exponential spreadsheet complexity v problem complexity relationship</td>
</tr>
<tr>
<td>---</td>
<td>---</td>
<td>---</td>
<td>---</td>
</tr>
<tr>
<td>7</td>
<td>Poor development tools</td>
<td>Lack of user demand, tool builder complacency, and risky economics</td>
<td>Development time longer than need be, errors easy to add hard to find</td>
</tr>
<tr>
<td>8</td>
<td>Poor quality VBA</td>
<td>Poor use of (freely and easily available) developer training</td>
<td>VBA is often more of a burden than an enabler</td>
</tr>
<tr>
<td>9</td>
<td>Ad-hoc nature of spreadsheets</td>
<td>Commercial pressure</td>
<td>Difficult to maintain, enhance and test.</td>
</tr>
<tr>
<td>10</td>
<td>Dangerous use of links</td>
<td>Quick reuse of previous results</td>
<td>Results may be inconsistent and or unrepeatable</td>
</tr>
</tbody>
</table>
4. OTHER FACTORS

Many researchers propose extra tools, methodologies, or training to impart some structure and robustness into spreadsheets and spreadsheet use. They miss several key facts:

1. People use spreadsheets **because** of their flexibility, not in spite of it.
2. Most people already have a robust tool for building structured models on their desktops. It’s called Microsoft Access, and most people ignore it because it’s not flexible enough for them.
3. Behind every spreadsheet horror, there is a deadline driven manager who prioritizes information timeliness over accuracy. In the modern commercial world where competitive advantage can last minutes (or less), wrong information is better than no information (as long as it’s not too wrong!)

If spreadsheets are so fragile and error prone why is so much work done with them? Cost, speed of development and current skill set. Excluding these 3 factors spreadsheets are probably never the right tool. But who can exclude these factors?

It has been suggested that spreadsheet use or abuse is an organisational thing (Cleary, 2004), commercial experience backs this up.

Spreadsheet use creates a web that quickly develops in uncontrolled environments (500 new spreadsheets per year (net of deletions), per analyst in one organisation (approx 2Gb of data suggesting an average size of 4Mb per model (typically 20-30 worksheets, 40,000 non blank cells per workbook) equates to 20 Million new data items per year (per analyst)).

Spreadsheets make a superb requirements development tool, and a great prototyping tool, but as every software textbook or developer will tell you, you must throw the prototype away. If you are building a racing car you wouldn’t start with a go cart, but you may make a clay prototype to test the aerodynamics.

5. THE FUTURE

Spreadsheets are the new legacy system with many organisations managing down their reliance on spreadsheets. Total replacement projects have had limited success. Technical solutions do not fix cultural problems.

A stronger business school focus on commercial database use rather than spreadsheets would prepare students for the modern work world of data manipulation rather than creation. The amount of information available electronically now is worlds away from what was available even 5-10 years ago.

Microsoft has woken up to the problems and potential of Excel. Expect to see a lot of work in this area as Microsoft attempt to leverage their ownership of the corporate desktop.
6. CONCLUSION

Errors or quality must be related to ‘fit for purpose’, and many commercial spreadsheets are probably just about good enough. A surprisingly large margin of error would not be catastrophic in many models. This tolerance is demonstrated by the lack of wholesale collapse of spreadsheet addicted organisations. Spreadsheet models are only one of the sources of information available and other sources may carry more weight.

Bad spreadsheets are a symptom not a cause. To address the problems associated with spreadsheets, the culture of spreadsheet abuse must be addressed.

Spreadsheets are a superb tool with many valuable uses. They do have limits however, and are frequently misused and abused. Blaming spreadsheets for the commercial challenges they introduce makes as much sense as blaming cars for car crashes. There is the occasional mechanical problem, but far and away the biggest culprit is operator error.

If you use spreadsheets you should know their limits as well as your own. Enter the spreadsheet maze at your own risk and with your eyes open. You have a choice of tools, choose wisely.

References


McConnell, S, (2004), Code Complete, Microsoft


18 Remediation Services for Excel, a comprehensive approach

Freek Varossieu, Cheryl O’Neil, Jan Pruis
Scientific Software Inc., Steenpad 19, 4797 SG Willemstad, The Netherlands,
freek@scisw.com

ABSTRACT

While auditors are enforcing the new regulations, companies turn to scrutinize their financial reporting with Excel. Their attention is drawn to the content of a spreadsheet and questions are asked regarding the authenticity and reliability of the data displayed. With open spreadsheets there is no control on the content and adequate measures need to be taken. Applying closed monitoring of content changes is a good and solid approach to ensure data integrity. Providing a user interface to display this cell-by-cell audit trail allows quick access to the spreadsheet’s history and gives auditors a ‘peace of mind’ about the data.

However, closed content monitoring on its own is not sufficient to ensure Excel integrity. As laid out in guidelines, access control and user roles must be in place. Many are trying to solve this with the normal NTFS security provided with the Microsoft operating systems. But this approach fails where access needs to be audit trailed due to tighter controls.

This article describes a solution that addresses both access control issues as well as close monitoring of spreadsheet content changes. This proven technology is already in place for several years and has been initiated by the tighter pharmaceutical industry controls. It is a simple and effective tool to meet the SOX section 404 guidelines.

Background

The introduction of the Sarbanes-Oxley regulations and later the Corporate Governance and Code Tabaksblat (corporate governance derivative) in The Netherlands) initiated a flurry of activities amongst companies. Originally estimated a 2.5 billion US dollar cost, the amounted investment by companies in SOX nears a 35 billion $. Little is spent on technology solutions that provide a solid foundation for compliance with aforementioned guidelines. Although technology alone will not provide compliance, a careful chosen product with the right implementation does help companies more than endless reports.

Placing the focus on Excel, a good reference document has been published by PriceWaterhouseCoopers in July 2004. They identified the following requirements for spreadsheet control:

1. Change Control
2. Version Control
3. Access Control
4. Input Control
5. Security and Integrity of Data
6. Documentation  
7. Development of Lifecycle  
8. Back-ups  
9. Archiving  
10. Logic Inspection  
11. Segregation of Duties and Roles  
12. Overall Analytics  

It is clear that point 1, 2, 3, 5, 6, 7, 8, 9 and 11 can be established by applying a central repository and necessary procedures. Workflow management is required for point 10. Controls on the content of a spreadsheet are mentioned in point 4 and 5.

Focusing only on managing the content of a spreadsheet will not result in SOX compliance. It is the overall approach that is required. It should therefore be recognized that integration between spreadsheet content auditing and access control is required.

**Proposed solution**

Keeping the overall approach in mind, the proposed solution is a combination of a central repository or Enterprise Content Management System (ECMS) and an Excel plug-in or Remediation Services for Microsoft Excel (RSME).

The central repository of ECMS replaces the, now often used NTFS security, and augments that with access control, versioning, audit trail, documented job types, virtual directory structure and a MD5 hash for each spreadsheet stored to ensure the authenticity of the record. With the secured access control one can grant and revoke access to certain spreadsheets. Combined with job types (user permissions) one can easily create a fine grid that maps to the actual organisation. The embedded audit trail functionality keeps track of who is entering the system and what the user is doing, e.g. which Excel spreadsheets are used and updated. This audit trail can be reported on request. Although trivial on first sight, a MD5 hash is used to uniquely identify each and every record and is paramount in version control. Each time a user saves a spreadsheet, the ECMS will create a new version, not overwriting the previous one. In this way older spreadsheets remain accessible.

Embedded lifecycle management and archiving functionalities complement the controls that are available to manage the spreadsheets. Life cycle management allows organizations to document and automate their procedures, ensuring that the appropriate steps are followed. Tracing of user activities provides documented evidence that the appropriate procedures have been followed. The support of electronic signatures simplifies the development lifecycle and logic inspection of spreadsheets.

Indexing of records offers easy search and retrieve of records. Often organizations get penalized not because they don’t have the required information, but simply that they can’t find it on-time. The ECMS allows automated meta-data extraction and even full text indexing. This simplifies the search and retrieve of spreadsheets.

As an Excel plug-in, RSME provides closed monitoring of the spreadsheets content. This cell-by-cell audit trail monitors not only value changes, but also function and macro changes. Stored in an embedded and encrypted database within the spreadsheet, this audit
trail is not accessible by normal means and cannot be excised. The Audit Trail on the content of the spreadsheet can only be viewed by the plug-in and provides search and filtering functionalities. If requested, the audit trail can be exported to different formats or printed. This audit trail is invoked at time of spreadsheet saving and prompts the user for a reason of change. The actual audit trail is a background process and does not require further user interference. This allows organizations to quickly move heir spreadsheets in a controlled environment without disrupting the day to day work of the employees.

To investigate the differences between spreadsheets, a compare functionality is included. This comparison functionality allows quick inspection of content changes between versions of a spreadsheet or between different spreadsheets. It allows to target for certain regions of interest or to zoom in on changed values and is a valued tool for documenting differences.

Protection of spreadsheets is now often done by applying a password to the spreadsheet. This is OK as long as the number of spreadsheets is manageable. An improved security is based on the domain user account as place an encrypted password on the spreadsheet.

Overview functionalities

As a stand-alone Excel plug-in RSME delivers a comprehensive audit trail on content changes. This audit trail is visible and searchable, allowing auditors to probe the data’s history. Adding ECMS as a backbone to manage the Excel spreadsheets allows tighter control and improved security. Development lifecycle, logic inspection and documentation are available and allow organizations to document and automate their processes. These would otherwise be manual activities, difficult to track and to documented and could easily lead to a ‘red tape’ approach to the problem.

Comprehensive overview of functionalities

<table>
<thead>
<tr>
<th>Control</th>
<th>ECMS + RSME</th>
<th>RSME stand-alone</th>
</tr>
</thead>
<tbody>
<tr>
<td>Access Control</td>
<td>√</td>
<td></td>
</tr>
<tr>
<td>Lifecycle management</td>
<td>√</td>
<td></td>
</tr>
<tr>
<td>Version control</td>
<td>√</td>
<td></td>
</tr>
<tr>
<td>Workflow</td>
<td>√</td>
<td></td>
</tr>
<tr>
<td>Documentation management</td>
<td>√</td>
<td></td>
</tr>
<tr>
<td>Domain Security</td>
<td>√</td>
<td></td>
</tr>
<tr>
<td>Spreadsheet locking</td>
<td>√</td>
<td></td>
</tr>
<tr>
<td>Closed Audit Trail</td>
<td>√</td>
<td></td>
</tr>
<tr>
<td>Spreadsheet compare</td>
<td>√</td>
<td></td>
</tr>
<tr>
<td>Archiving</td>
<td>√</td>
<td></td>
</tr>
<tr>
<td>Back-up</td>
<td>√</td>
<td></td>
</tr>
<tr>
<td>Roles/duties</td>
<td>√</td>
<td></td>
</tr>
<tr>
<td>MD5 hash</td>
<td>√</td>
<td></td>
</tr>
<tr>
<td>Indexing</td>
<td>√</td>
<td></td>
</tr>
<tr>
<td>Search/retrieve</td>
<td>√</td>
<td></td>
</tr>
</tbody>
</table>
Conclusion

Providing SOX compliance for Excel spreadsheets cannot be seen as a stand-alone activity. Adding closed monitoring to spreadsheets is a part of a more holistic approach to SOX compliance. The vast majority of activities are centered on the management of the record itself. Combining an ECMS with RSME will improve SOX compliance.

References

PriceWaterhouseCoopers (June 2004), The use of spreadsheets: considerations for section 404 of Sarbanes-Oxley act.

Scientific Software (February 2005), Effective risk management for financial spreadsheets
Archiving: The Overlooked Spreadsheet Risk
Dr. Victoria Lemieux
IT Risk & Security
Credit Suisse First Boston
One Cabot Square, London, E14 4QJ
Vicki.lemieux@csfb.com

ABSTRACT

This paper maintains that archiving has been overlooked as a key spreadsheet internal control. The case of failed Jamaican commercial banks demonstrates how poor archiving can lead to weaknesses in spreadsheet control that contribute to operational risk. In addition, the Sarbanes-Oxley Act contains a number of provisions that require tighter control over the archiving of spreadsheets. To mitigate operational risks and achieve compliance with the records-related provisions of Sarbanes-Oxley, the author argues that organisations should introduce records management programmes that provide control over the archiving of spreadsheets. At a minimum, spreadsheet archiving controls should identify and ensure compliance with retention requirements, support document production in the event of regulatory inquiries or litigation, and prevent unauthorised destruction of records.

1. INTRODUCTION

Many companies rely on spreadsheets for financial reporting and support of operating processes. For example, in a financial services firm, spreadsheets may be used to perform reconciliations by downloading information from two systems into separate existing MS Excel spreadsheets. MS Excel functions and pivot tables are then used to create summary data for each source. When spreadsheets are core to business processes poor control over them can have a significantly negative effect upon companies' bottom line and reputation. PriceWaterhouseCoopers reports the following examples of how spreadsheet risks can impact upon the corporate bottom line (PwC, 2005):

-A spreadsheet error at a major financial institution was deemed a significant factor in a major $1billion financial statement error in the classification of securities. The error resulted from a flawed change control process - an unapproved change to the formula within the spreadsheet -and other control deficiencies, including lack of technical and user documentation, insufficient testing, and inadequate backup and recovery procedures.

-A utilities company took a $24million dollar charge to earnings after a spreadsheet error -simple mistake in cutting and pasting - resulted in an erroneous bid in the purchase of hedging contracts at a higher price than it wanted to pay.

-A trader at a bank was able to perpetrate fraud by manipulating spreadsheet models used by the bank's risk control staff. Because of inadequate controls over spreadsheet this fraud continued for months.
Not surprisingly then, being able to demonstrate sound internal control of critical spreadsheets in compliance with the Sarbanes-Oxley Act (SOX) is as important as being able to demonstrate it in respect of core processing applications and other critical systems. With the introduction of SOX, applying sound internal controls to spreadsheets not only makes good business sense but also becomes a legal requirement.

This paper will argue that archiving has been overlooked as a key spreadsheet internal control. Using a case study of failed Jamaican commercial banks, it will demonstrate how poor archiving can lead to weaknesses in spreadsheet control that contribute to operational risk. This will be followed by a discussion of SOX archiving requirements and how to mitigate archiving risks and introduce into an organisation best spreadsheet archiving practices for SOX compliance.

2. ARCHIVING: THE OVERLOOKED SPREADSHEET RISK

Discussions of spreadsheet-related risks generally focus on:

- Complexity of the spreadsheet and calculations
- Purpose and use of the spreadsheet
- Number of spreadsheet users
- Type of potential input, logic and interface errors
- Size of the spreadsheet
- Degree of understanding and documentation of spreadsheet requirements by the Developer
- Uses of the spreadsheet's output
- Frequency and extend of changes and modifications to the spreadsheet
- Development and developer training and testing before the use of the spreadsheet (PWC 2005).

Equally important, however, are the risks associated with failing to properly archive spreadsheets.

Why should spreadsheet archiving be considered a critical risk area? Simple: there are risks to the business when critical information is not properly retained and accessible, especially in the post-SOX world.

3. A CASE STUDY OF ARCHIVING RISKS

A study of the Jamaican banking crisis, in which all of the country's indigenous commercial banks failed, shows how poor control of spreadsheet archiving contributed to the flawed decision making that, in turn, fed into to a failure of Jamaican banks (Lemieux, 2002). Like many other firms, the banks relied heavily on spreadsheets because their major transaction processing and risk management systems failed to meet management information and reporting requirements.

Interviews with former employees of the failed banks reveal that they used spreadsheets in the following ways:

- Cash management
- Financial control and budgeting
Archiving: The Overlooked Spreadsheet Risk
Dr Victoria Lemeieux

- Analysis of customer and product profitability
- Analysis of the cost of funds
- Currency position management
- Credit decision management
- Interest rate sensitivity analysis and risk management
- Recording of proprietary trading in securities.

There was an ad hoc approach to spreadsheet archiving at these banks, characterised by:

- Individualistic naming of files. Individuals were allowed to assign their own names to files. These names often gave no clue as to the content of the file or its relation to a business process. Ultimately, individualistic naming of files was a major factor in the inability to locate important spreadsheets. Once the creator of the spreadsheet left a bank, the spreadsheet was as good as gone with them since the knowledge of its existence and how to retrieve it vanished with the individual who named and stored it. Even when the creator was still around, individually named spreadsheets became "information islands", often only available to the single user responsible for their creation even though the information they contained was of benefit to the decision-making processes of others.

- Ad hoc assignment of storage location. Individuals were permitted to store spreadsheets in their personal drives to which they alone had access and for which they alone made decisions about retention or deletion of documents.

- Absence of any objective criteria governing deletion of spreadsheets from storage. Without any clear understanding of the importance of spreadsheets in the management information and reporting process, no one saw any reason to control their deletion. This resulted in periodic purges of important spreadsheets as storage locations became full. Since there was no understanding of these spreadsheets as "records" that needed to be kept as evidence of how the bank's financial positions were calculated, when drives filled up and the notice came round from IT to make more space available, individuals purged their drives and wiped out the evidentiary trail.

- Failure to preserve a link to the business context in which the spreadsheets were created. This failure often rendered spreadsheets meaningless as background to a particular business decision.

- Inability to guarantee the authenticity and reliability of spreadsheets. Since there were no controls over how spreadsheets were archived and no effort was made to "lock" down their content as part of a formal archiving process, if anyone was lucky enough to actually locate one of these documents after a period of time, their integrity was seriously questionable, since anyone could have changed the content in the intervening period and audit trail controls were weak to non-existent.

Former managers at the failed banks commented on the impact of poor spreadsheet controls on the banks. One interviewee, for example, described how the bank was forced to rely on competitors' actions to drive its asset and liability management (ALM) policies because the banks own interest rate sensitivity data, which had been recorded in spreadsheets, could no longer be accessed. (Lemieux, 2002, p.258).

Poor control over spreadsheets at Jamaican indigenous banks contributed to management information and external reporting problems (i.e., P&L distortions) that
Archiving: The Overlooked Spreadsheet Risk
Dr Victoria Lemeieux

contributed to the banks' management and external regulators losing sight of the banks' true positions and exposures. This problem fed a downward spiral into liquidity crisis.

As the Jamaican financial crisis unfolded, the government also recognised that fraud and corruption had contributed to the collapse of indigenous banks. To address these allegations, it established a team of foreign and local forensic auditors to work with the police fraud squad to identify and take action on instances of fraud. Inaccessibility of source documents, however, seriously hampered the auditors' work. The work of reconstructing what in some cases were very convoluted financial transactions was made extremely difficult by the fact that critical records, many in spreadsheet form, were missing. One interviewee said: "I am looking at a particular company now where I thought that I was told that all the servers there that I could run off the information. When somebody attempted to do that they realised that the diskette was bad or contaminated. So you have a whole year's [data] that you cannot access . . . I have tried so all I have now left to do is to utilise some of the hard copies. But it is not consistent. You have one month, you can't find two months and you have another month. So it is going to be very difficult to trace these transactions." (Lemieux, 2001, p.338).

Though the absence of archival controls over spreadsheets at the failed indigenous banks may have been extreme, similar problems are not unknown in other firms. Though no separate data exist for spreadsheets, the aggregated data covering all electronic records indicates that very few organisations have established formal programmes to systematically manage the archiving of their electronic records (A11M, 2005). Many still rely on back up processes better suited to disaster recovery than to the preservation of evidence to meet legal and regulatory requirements.

4. WHAT SOX SAYS ABOUT ARCHIVING

In the post-SOX world control over the archiving of spreadsheets becomes a critical compliance matter as well as a business competitiveness issue. There are a number of SOX provisions that impact upon records and information management, some more relevant to the matter of archiving spreadsheets than others. This section will highlight just a few of these (a full list of the requirements will be available in the appendix to the published version of this paper):

1. SOX 103(aX2)(A)(i) - Audit reports, work papers, and other information related to any audit report must be kept for at least 7 years; Audit reports must contain statements about the testing of internal controls and whether those control structures include maintenance of accounting records.
2. SOX 104(e) - Public accounting firms may be required to retain records not otherwise required under section 103.
3. SOX s. 802, rule 2-06(a) requires a 7 years after conclusion of the audit/review retention period for accountants to retain "records relevant to the audit or review of issuers' and registered investment companies' financial statements, including work papers and other documents that form the basis of the audit or review, and memoranda, correspondence, communications, other documents and records (including electronic records), which are created, sent or received in connection with the audit or review." A company, in consultation with its legal and accounting advisors, will need to determine which of its spreadsheets and other records fall within the meaning of this provision of the act and related rules. There is a 10-year penalty for violating this rule.
Given the records and information management requirements under SOX, express or implied, and the penalties for non-compliance, poor control over spreadsheet archiving is a risk that should not be left unmitigated.

5. ADDRESSING SPREADSHEET ARCHIVING RISK - ELEMENTS OF GOOD PRACTICE

It is best to address spreadsheet archiving as part of setting up (if there is no programme in place), maintaining, and ensuring compliance with an organisation-wide records management programme. This approach also will demonstrate that archiving controls are part of business as usual practice, not merely a "tick the box" approach to SOX compliance. It also will ensure that SOX controls do not work at cross-purposes with other organisational records requirements and controls.

SOX does not explicitly direct any records management activities, so organisations cannot simply follow a statutory recipe to achieve good archiving practices. However, the Act demands a number of specific outcomes that need to be underpinned by effective records and information management practices. The requirements of an organisation's records management programme should be guided by what is needed to meet these outcomes. International standards, such as ISO 15489, the International Records Management Standard, can provide guidance on how to go about establishing a compliant records management programme (ISO, 2001). At the end of the day, the goal of the records management programme should be to create the processes, procedures and records necessary to demonstrate compliance with SOX and to repudiate any claims of misfeasance or malfeasance (Montana et al., 2003).

6. RETAINING RECORDS

Good records management practice calls for the establishment of Records Retention Schedules. These are documents that identify the records that must be created by law or regulation, and the period of time for which those records must be retained. An organisation should definitely have one of these documents in place.

Often, one finds that an organisation's archiving function has established a Records Retention Schedule, but that the scope of its coverage only extends to paper documents. There is no question that SOX requirements apply not just to paper records, but also to documents in a multitude of electronic forms, including electronic versions of spreadsheets. Indeed, ever since electronic forms of documents have become ubiquitous, the U.S. courts have shown a distinct favouritism for the submission of evidence in its "native" form (i.e., electronic) rather than receiving a paper "copy" (Wallace, 2001). Consequently organisations should be clear that their Records Retention Schedules apply to records in all forms.

In terms of the retention requirements related to SOX, there seems to be much debate and confusion in this area. Recent discussions on the IT Governance listserv led to a wild claim that all records had to be retained for 7 years. This is not the case. Sections 103(a)(2)(A)(i) and 802(1)(a) apply to audit records and audit work papers of public accounting firms. Industry best practice has evolved to include internal audit records and work papers as well, though there is no explicit requirement for retention of these records in the Act or its related regulations. Initially, the Act required that audit work papers be kept for a period of 7 years under section 103 and 5 years under section 802. The requirements under section 802 have subsequently been raised via U.S. Securities and Exchange Commission regulation to 7 years in order to harmonise with
section 103 of the Act and with auditing standards (SEC, 2003). However, the Act also says that (s.802(2)(c)) nothing in it should be taken to diminish or relieve an obligation to comply with the records retention requirements or prohibitions on document destruction mandated by other legislation. This means that if audit records fall within the retention requirements of other legislation and those retention requirements are longer, the longer of the requirements would apply.

Audit-related records are the beginning and end of explicit retention requirements in SOX, but clearly the letter of the Act requires compliance with the retention requirements of other legislation (s. 802(2)(c)above). Moreover, the requirements of section 404 are underpinned by evidence of the establishment and proper operation of effective internal controls. This implies a much wider obligation on organisations to retain records. Many of these records will be spreadsheets created as part of SOX-relevant business processes. Though the Act focuses on the accuracy of corporate financial records, an organisation would be foolish to stop its records management efforts with financial records. The fact is that non-financial records can provide evidence of financial vulnerabilities. As such they will be deemed relevant to any SOX-related requirements and inquiries.

Another SOX-records retention myth that needs to be explored and, in my view, exploded, is that SOX-relevant records must be gathered up and kept in a single, SOX records repository. This would certainly be one approach to ensuring retention of the records required by SOX. The expense, however, could be prohibitive. Aside from the expense of gathering up all SOX-relevant records for retention, there is the question of whether removing the records from their business context has the potential to diminish the evidentiary qualities of the records. Unless carefully procedurally controlled, there easily could be a danger of reduced record integrity. Therefore, a better approach is to identify the records, properly manage and archive them "in situ" (i.e., within a production environment) or a corporate archiving environment, and apply appropriate indexing for retrieval.

Having said that Records Retention Schedules apply to records in all forms, including spreadsheets, the trick is putting them into effect. Most organisations retain electronic spreadsheets, as in the Jamaican case study above, on a variety of servers, and leave control over the life cycle of such documents to the individuals who generated them in the first place (i.e., usually an end user). In the post-SOX environment, it should be abundantly clear that this approach is no longer advisable.

Steps must be taken to ensure that spreadsheet content, structure and context, that is the links to the business transactions that they were created to support, are retained for their required period of time in a form acceptable to regulators, investigators and the courts. This paper will return in a later section to the implementation of Records Retention Schedule; but for now, suffice it to say, regulators have shown a definite impatience with organisations that are not able to produce requested documentation. For example, in 2002 the U.S. Securities and Exchange Commission levied fines against five investment banks for failure to preserve emails (SEC, 2002).

Records Retention Schedules must not only be implemented, they must be regularly reviewed. A regular review will ensure that the Records Retention Schedule remains consistent with legal and regulatory requirements, complete (i.e., incorporates the records generated from new business functions), and appropriate to the business environment. When reviewing Records Retention Schedules, therefore, organisations should look at the currency of its retention periods, records series, nomenclature, indexing and structure and overall compliance. A regularly reviewed Records
Retention Schedule will be a good defence strategy in the event of the kind of scrutiny that records management programmes can now come under in the event of SOX-related investigation or litigation.

7. RETRIEVING RECORDS

Documentation must be capable of being accurately and quickly retrieved in the event of an investigation, regulatory inquiry or litigation. Even the most complete documentation loses value if it cannot be retrieved. Given the climate of suspicion that was the impetus of the Act, delays in the production of legitimately requested records and information can be extremely damaging. Opponents and, it should be added, the public, are willing to assume bad motive and push for sanctions. For example, recently, a Florida court penalised the investment bank Morgan Stanley for "bad faith" actions in respect to handing over backup tapes containing emails relevant to the Perelman litigation against Sunbeam. The judge told the jury it should simply assume that Morgan Stanley helped defraud Mr. Perelman (Craig, 2005). Even without regulatory sanction or court-imposed penalties, the reputational damage can be significant.

The ability to accurately and quickly retrieve documentation will be assisted by setting up standardized file structures, implementing file naming conventions, and indexing spreadsheets. Personnel must also be sufficiently well trained to carry out document requests efficiently and in the time frame demanded, as well as to understand the need to protect the integrity of the documents throughout the retrieval process.

File naming conventions deserve special attention because one of the most common failings of retrieval systems is due to poor nomenclature. Under the best of circumstances, poor nomenclature impedes the ability of users to retrieve information efficiently. Under the worse case scenario, it can be interpreted more sinesterly. Poor naming conventions may be taken as an attempt to conceal information or, as in the recent Citigroup European Bond Trading Scandal wherein the bank's highly contentious bond trading move was rather ominously and unfortunately named "Dr. Evil" (Wall Street Journal, 2005), arm opponents and cause damage to a Firm's reputation. Clarity and transparency should be key goals in the development of file naming conventions and indexing plans.

Many organisations fail to preserve the links between individual documents and the business context to which they relate. This can be a mistake as it can render a document difficult to locate, open the meaning of the document up to "creative" interpretation by adversaries, and render it difficult to determine whether a document legitimately falls within the scope of a document production order or legal discovery exercise. Classification of documents according to corporate records taxonomy can serve as a vehicle for preserving contextual links in documents. Another means of achieving this goal is to capture contextual metadata, such as the name of the business process or transaction for which the document is being created, which is either stored in a database or embedded in the document itself. Business process flows can be another very useful way of capturing information about how spreadsheets fit into the overall business context.
8. DESTROYING RECORDS

One might think, given the harsh penalties associated with records destruction, that, record management programme or no records management programme, any records destruction should be halted. Quite to the contrary, records destruction should still take place in keeping with best practice, but it must be as part of the normal and ordinary course of business. The best way to demonstrate that legitimate destruction is for records disposals to take place in the context of an established records management programme and with full audit trails of disposal actions. In contrast, for any organisation undergoing Sarbanes-Oxley scrutiny, ad hoc destruction of records in the absence of a formal programme, no matter how innocent the motive, invites the most damning inferences as to reasons." (Montana et al., 2003).

ISO 15489, the International Records Management Standard, establishes the following principles governing records disposal (ISO, 2001):

- Disposition authorities that govern the removal of records from operational systems should be applied to records on a systematic and routine basis in the course of normal business activity.

- No disposition action should take place without the assurance that the record is no longer required, that no work is outstanding and that no litigation or investigation is current or pending (or even reasonably foreseeable) which would involve relying on the records as evidence.

- Destruction should always be authorised.

- Records pertaining to pending or actual litigation should not be destroyed.

- Records destruction should be carried out in a way that preserves the confidentiality of any information they contain.

- All copies of records that are authorised for destruction, including security copies, preservation copies and backup copies, should be destroyed.

- Records systems should be capable of facilitating and implementing decisions on retention or disposition of records.

- It should be possible for these decisions to be made at any time in the existence of the records including during the design stage of the records systems.

- It should also be possible, where appropriate, for disposition to be activated automatically.

- Systems should provide audit trails or other methods to track completed disposition actions.

Given the penalties in the Act associated with destruction of records, it is worth focusing some detailed attention on the subject of destruction bans and legal holds on records destruction. The Act has established a Public Company Accounting Oversight Board (PCAOB). The PCAOB is vested with broad power to oversee public
accountancy, set standards for the conduct of audits and maintenance of records by public accountants, and generally to oversee and enforce standards of public accounting.

The parties most directly affected by the PCA013 are public accountants and auditors, but the PCA013 also has investigative authority over the auditing of public companies. In general, the PCAOB is empowered to request and/or subpoena documents in the possession of any person, including a client of a registered public accounting firm, which the Board considers relevant or material to an investigation. A publicly traded company, thus, may find itself required to respond to an investigation by the PCAOB by the production of documents and information related to an audit. It is therefore imperative for organisations to have in place policies, procedures and systems for handling information production demands arising out of government investigations, litigation, and other legal and adversarial situations. Even for organisations that already have such policies, procedures and systems in place, the Act stipulates requirements that should encourage a check for efficacy in relation to the following (Montana, et al, p. 18):

- The Act grants authority to demand production of testimony or documents well in advance of any formal proceeding such as litigation. Are procedures sufficient to ensure that documents are safeguarded from the moment such proceedings are reasonably foreseeable?

- Are existing procedures sound and foolproof? For example, once a document destruction hold order has been issued, is the organisation confident that no documents will be destroyed. In many circumstances, implementation of a document destruction hold is the responsibility of the employee and there are very few controls in place to ensure this responsibility will be carried out.

- Is staff training and awareness sufficient to provide documents in the timeframe and with the accuracy required?

9. ARCHIVING STRATEGIES: SOME PROPOSALS

Having discussed general good practice in respect to archiving, this paper will now turn to discussing the specifics of how these practices might be applied to spreadsheet archiving.

SOX essentially requires that spreadsheets be dealt with as any other record that would be required as evidence to substantiate an organisation's financial statements. All records have a life cycle consisting of the following phases:

1. Creation and/or receipt
2. Active use
3. Semi-active use, during which records are referred less often because the business transaction for which they were created or received has been completed.
4. Inactive use, during which records are rarely referred to but must be retained for legal/regulatory or business reasons. It is during the inactive period that records generally are migrated from production environments to an archive and/or deleted from production environments.
The records life cycle roughly parallels the software life cycle. As with other types of records, effective spreadsheet archiving will begin at the point of spreadsheet creation and end only when the spreadsheet has met all retention requirements.

Like Word documents or MS Access databases, spreadsheets are created using end-user processing technology readily available on the desktop. As such, spreadsheets are often created and managed by the end-user, who may be very unfamiliar with the principles of managing the software or records life cycle. In some cases, however, because the spreadsheet performs quite complex processing functions and forms a critical bridge between applications in key business processes, an organisation's IT department may become involved in the design of the spreadsheet or aspects of its management. For this reason, it can be helpful to classify spreadsheets into two broad categories as follows so that the archiving strategy can be tailored to the level of end-user control and processing complexity associated with the spreadsheet. The following two categories are recommended:

1. Spreadsheets that do not, or only minimally, process data and which are created and maintained by end users, and

2. Spreadsheets that do more complex processing of data in order to perform or support critical processes and in which the IT department may have more involvement in the design and management.

Some examples based on the reconciliation of trading transactions will serve to illustrate what types of spreadsheets would fall into these two categories. Spreadsheets not used to process data include spreadsheets in which an individual compares reports from two systems and records any breaks as the list of exceptions for a particular day. Also included in this category would be a spreadsheet in which information is downloaded from a system and in which a pivot table is then used to create summary data. On the other had, spreadsheets that are used to process data would include a scenario in which information is downloaded from two systems into two separate MS Excel spreadsheets. MS Excel functions and pivot tables are then used to create summary data for each source. The data is then manually reconciled.

It is recommended that the archiving of spreadsheets that fall into the first category be dealt with in the same manner as the archiving of other unstructured content (e.g., Word files, some MS Access databases). Spreadsheets that perform more complex processing functions, on the other hand, are better handled as mini applications in which their archiving is dealt with in the context of managing the application life cycle.

10. ARCHIVING SPREADSHEETS AS UNSTRUCTURED CONTENT

Although, spreadsheets that fall into the first category of spreadsheet do not perform complex functions, they still do form an important part of the trail of evidence that SOX requires. For this reason, it is risky simply to rely on archiving the source data and recreating the spreadsheet in the event of a request for documentation, as investigators will be looking for evidence with integrity and authenticity (i.e., documentation produced contemporaneously in the normal and ordinary course of business). As such, it is a much less risky strategy to preserve, and be in a position to present, the entire evidentiary trail - source data and spreadsheet.
Generally speaking the creation and management of spreadsheets, that fall in the first category falls to the end user. To ensure that an organisation does not develop the problems experienced by the failed Jamaican banks, an organisation should establish some controls over how end users create and store spreadsheets. The following offers some examples of scaleable archiving strategies that an organization can pursue to implement spreadsheet archiving controls for spreadsheets that do not perform complex data processing:

1. Low criticality/small scale operations - designate folder on server as archival folder and place all spreadsheets in folder in P1317 or PDFIA format to lock down content. Naming of files in the folder should be standardised, and documented controls should be established over who does the archiving, who has access to the folder, who can delete, etc. Ensure no deletions of files that fall within business critical/SOX relevant categories before their required retention period has been fully met or of any files if relevant to an ongoing or reasonably anticipated investigation, etc. Keep all files in online storage until retention period is expired, or if using removable storage media such as tape, establish a formalised programme to regularly review its integrity and refresh the medium or migrate content as necessary. Note: virus check before you put anything into your archival store to protect your archival records.

2. High criticality/large scale operations. Introduce an electronic document management system with WORM storage. Institute Information Life Cycle Management (ILM) processes. The proper operation of EDRMS depends upon having a well-thought out and constructed corporate taxonomy, as it is the taxonomy which identifies the categories of business records that that the organisation creates and received, and the retention requirements that apply to each type. Organisations may want to look at taxonomy management software to support this, as taxonomy development and management can consume a large amount of resource. When an EDRM system is used to support spreadsheet archiving, end users (or the system if using an auto-classification feature) will associate the spreadsheet to an appropriate category in the taxonomy thereby ensuring that the spreadsheet will be retained for the period of time indicated by its association with a particular category.

3. Med. Criticality/med scale operations. Mix of 1 and 3

11. ARCHIVING SPREADSHEETS AS SOFTWARE ASSETS

Spreadsheets that fall into the second category of spreadsheet (i.e., those that perform more complex processing) may or may not be developed and/or supported by an IT department, depending on the organisational context. Regardless, given the function they perform, these spreadsheets may be treated more suitably as mini applications and their archiving dealt with in the context of managing the software life cycle.

Paying attention to data archiving requirements at the time at which a spreadsheet is created can make the process of archiving much easier and more effective in the long run. This is much easier to do if one applies a system development life cycle approach to the development of complex spreadsheets. It can be very useful to insert a records retention checkpoint at the system development project initiation phase. For example,
Spreadsheet designers could be asked whether they have identified the retention requirements for the spreadsheet and to outline how those retention requirements will be met. This will alert spreadsheet designers to the necessity of considering retention requirements and help them plan for data archiving.

Spreadsheet designers will be supported in their efforts if they can refer to organisation wide Records Retention Schedules that identify the retention requirements for given types of data and if the organisation has established standard data archiving solutions.

12. MIGRATING SPREADSHEETS OUT OF A PRODUCTION ENVIRONMENT TO AN ARCHIVE

It was once the case that organisations relied on backup tapes for both disaster recovery and retention purposes. It is now generally agreed that reliance on backup processes is no longer a suitable data archiving strategy. The problem with a reliance on backup processes, according to a Robert Frances Group research note is that administrators spend up to six hours per week recovering old messages for users, and responding to legal discovery can cost hundreds of thousands of pounds (RFG, 2005). Since the need to catalogue, locate and retrieve information in a timely manner has become more urgent in large part due to regulatory pressure, organisations have begun to embrace information life cycle management (ILM) in order to be able to free space for mission-critical data and provide an index and audit trail of archived information to support corporate governance.

This has given risen to a need for archiving tools that perform a long-term information and preservation access function; are able to keep up with steady input streams as well as that have period inputs; allow a wide variety of organisational arrangements (i.e., local implementations covering inputs from one production system or a central archives covering inputs from several production systems). In response, software vendors have begun to offer archiving tools that can be used to handle the archiving of more complex and critical spreadsheets. These archiving tools are designed to take data from a production environment and migrate it to off-line storage for retention until that data is no longer needed. Compliance oriented archiving tools supply comprehensive storage, storage management, and security offerings to address data retention needs. In addition, a number of third-party archiving providers have emerged, so an organisation need not maintain its own archive.

When archiving spreadsheets to off-line and less expensive storage, the spreadsheet can be retained in its production format (e.g., an MS Excel file) or it can be converted to and retained in an open standard format such as PDF/A or WL to protect against technological change. The decision about the best format in which to retain the file should be in proportion to the expected length of time for which the spreadsheet must be retained, that is, the longer the retention, the better it will be to retain in an open format.

In terms of managing the migration of spreadsheet data from a production environment to a data archive environment, ISO 14721, the Open Archival Information System Standard, presents a reference model for the preservation of data that provides very useful guidance. It should be noted that the migration of data into archival systems needs to be tightly controlled in order to ensure that data is not lost and that data authenticity and integrity is maintained. For the same reason, control
must be maintained over the management of the archival repository at all times. Here again, ISO 14721 is very instructive on this point.

12. CONCLUSION

Most discussions of spreadsheet risk focus on the factors contributing to accuracy and reliability of spreadsheet data content. Archiving, however, is often overlooked. But, as argued in this paper, it is critical for full SOX compliance. While SOX does not provide detailed guidance on corporate record keeping, the absence of such guidance should not be taken to mean that an organisation's leadership would not be expected to assess whether they have potential vulnerabilities and liabilities as a result of poor spreadsheet archiving practices and to take step to mitigate these. In their publication on the records management of Sarbanes-Oxley, authors John Montana, J. Edwin Dietal and Christine Martins write: "All corporate recordkeeping is going to be under much closer and intense scrutiny in the future... One must be able to show that an aggressive, thoughtful, innovative corporate records and information management program is in place and continually being improved to ensure that individuals that might allege failure to comply with Sarbanes-Oxley are not successful." (Montana, et al., 2003)

REFERENCES


APPENDIX A
RECORDS-RELATED REQUIREMENTS IN THE SARBANES-OXLEY ACT

4. SOX 102(e) - Registration applications and annual reports must be available for public inspection subject to rules of the Board or Commission and applicable confidentiality laws.

5. SOX 103(a)(2)(A)(i) - The Board shall establish quality control and ethical standards for registered public accounting firms in the preparation and issuance of audit reports; Audit reports, work papers, and other information related to any audit report must be kept for at least 7 years; Audit reports must contain statements about the testing of internal controls and whether those control structures include maintenance of accounting records.

6. SOX 104(e) - Public accounting firms may be required to retain records not otherwise required under section 103.
   SOX 105(b)(2)(B)(C) & (D) - The Board may require production of audit work papers or any other documents in the possession of a registered public accounting firm or any other person, including any client of the firm.

7. SOX 105(b)(5)(A) - All documents and information prepared by or given to the Board, related to an investigation under section 104, including Board deliberations, are confidential. (With certain enumerated exceptions under paragraph (B).)

9. SOX 105 (c)(1) - The Board must keep a record of its proceedings.

10. SOX 105(c )(5)(A) & (B) - Applies sanctions to both intentional and negligent conduct.

11. SOX 106 - Foreign accounting firms that issue opinions or otherwise perform material services for a US company must supply audit work papers to the Board or Commission in connection with any investigation and be subject to the jurisdiction of US courts.

12. SOX 201 - Amends section 1 OA of the Securities Exchange Act of 1934 to prohibit registered public accounting firms from providing bookkeeping or other services related to accounting records or financial statements contemporaneously with audit services. It also precludes them from designing or implementing financial information systems at the same time, as well as performing other enumerated services contemporaneous with an audit.

13. SOX 202 - Amends section I OA of the Securities Exchange Act of 1934 to require Audit Committees to preapproval all audit and nonaudit services with certain enumerated exceptions.

14. SOX 204 - Amends section 1 OA of the Securities Exchange Act of 1934 to require accounting firms to report to Audit Committees all critical accounting policies and practices to be sued, all alternative treatments of financial information that have been discussed and other "material written communications" between the accounting firm and the management of the company.
15. SOX 301 - Amends section 1 OA of the Securities Exchange Act of 1934 to make Audit Committees establish procedures for the receipt, retention, and treatment of complaints regarding accounting, internal accounting controls, or auditing matters. Does not list a period of years for retention. Audit Committees to establish retention conditions.

16. SOX 306 - Prohibits any director or executive officer from purchasing, selling, or otherwise acquiring or transferring any equity security of the issuer during a blackout period if he/she acquires it in connection with his/her service or employment as a director or executive officer. Any profit realized by him/her will be recoverable by the issuer. Action to recover profits must be brought within 2 years of the date on which the profit was realized.

17. SOX 404 - Management Assessment of Internal Controls - many spreadsheets, will form a key component of being able to substantiate a company's financial statements and must therefore be available to the regulators if a company's statements are questioned. As noted by the U.S. Securities and Exchange Commission, increased retention of identified records also may provide critical evidence of financial reporting impropriety or deficiencies in the audit process." (SEC, 2003). Verification of a company's financial statements for a given period may take place a number of years after release, so a company should be prepared to produce supporting documents until certain the verification process is complete. There are serious penalties for making a false declaration - up to 10 years - so company directors and senior managers will want to be sure they can substantiate their financials.

18. SOX s. 802, rule 2-06(a) requires a 7 years after conclusion of the audit/review retention period for accountants to retain "records relevant to the audit or review of issuers' and registered investment companies' financial statements, including workpapers and other documents that form the basis of the audit or review, and memoranda, correspondence, communications, other documents and records (including electronic records), which are created, sent or received in connection with the audit or review." A company, in consultation with its legal and accounting advisors, will need to determine which of its spreadsheets and other records fall within the meaning of this provision of the act and related rules. There is a 10 year penalty for violating this rule.

19. SOX 802a - criminal penalties and their implications. "Whoever knowingly alters, destroys, mutilates, conceals, or makes a false entry in any record, document or tangible object with the intent to impede, obstruct, or influence the investigation or proper administration of any matter within the jurisdiction of any department or agency of the United States or any case filed under title 11, or in relation to or contemplation of any such matter or case, shall be fined under this title, imprisoned not more than 20 years, or both." This establishes the need for a legal hold regime so as to halt the deletion of any spreadsheet or other document that may be needed in case of litigation or investigation, even if only at the point of being anticipated.

20. SOX 906 - Requires written statements from CE0s and CFOs in annual reports to certify that the information is in compliance with the Securities Exchange Act of 1934 and that the report fairly represents the financial condition and results of operations of the issuer. "Criminal penalties of not more than $5,000,000, or
imprisonment of not more than 20 years, or both for wilfully certifying any statement, knowing that it does not comport with all the requirements set forth."

21. SOX 1102 - Which says that "Whoever corruptly (1) alters, destroys, mutilates, or conceals a record, document, or other object, or attempts to do so, with the intent to impair an object's integrity or availability for use in an official proceeding; or (2) otherwise obstructs, influences, or impedes any official proceeding, or attempts to do so, shall be fined under this title, or imprisoned for not more than 20 years, or both.

22. SOX 1106 - Amends section 32(a) of the Securities Exchange Act of 1934 to increase the penalties of section 78ff(a)

23. SOX 1107 - Amends section 1513 of the title 18 United States Code to punish retaliation against informants by fines or imprisonment of not more than 10 years, or both. The amount of possible fines is not stated in the Act.
This volume contains the proceedings of the sixth European conference on Managing Spreadsheets in the light of Sarbanes-Oxley, EuSpRIG 2005, held in July 2005 at University of Greenwich.

The objective of this conference is to promote discussion and cooperation amongst those concerned with authorising, auditing or developing spreadsheet models and by so doing, improve the reliability and integrity of information portrayed in spreadsheet models.

The papers cover a broad spectrum of practical experience and research.