10th EuSpRIG Annual Conference

Le rôle des tableurs pour l'excellence dans les organisations

The Role of Spreadsheets in Organisational Excellence

STEF ENS Cachan, Paris, 2-3 July 2009

Conference Proceedings
European Spreadsheet Risks Interest Group

EuSpRIG 2009 Conference

The Role of Spreadsheets in Organisational Excellence

Le rôle des tableurs pour l'excellence dans les organisations

Conference Co-Sponsored By:

Information Systems Audit & Control Association
(Northern England Chapter)
Baker Tilly Corporate Finance LLP
UWIC
University of Greenwich
Auditnet.org
Systems Modelling
Spreadsheet Engineering Ltd
ENS Cachan

Proceedings Editors:
David Ward & Grenville Croll

July 2009
Ecole Normale Supérieure de Cachan, Paris, France.

Proceedings of the EuSpRIG 2009 Conference
"The Role of Spreadsheets in Organisational Excellence" ISBN : 978-1-905617-89-0
Copyright © 2009, European Spreadsheet Risks Interest Group (www.eusprig.org)
EuSpRIG 2009 Conference

“The Role of Spreadsheets in Organisational Excellence”

“Le rôle des tableurs pour l'excellence dans les organisations”

EUSPRIG 2009 ORGANISING COMMITTEE

Pat Cleary (Chair), University of Wales Institute Cardiff (UK)
Eric Bruillard, STEF ENS Cachan, Paris, France
François-Marie Blondel, STEF ENS Cachan, Paris, France
Françoise Tort, STEF ENS Cachan, Paris, France
David Chadwick, University of Greenwich, UK
Ray Butler, Highways Agency, UK
Simon Thorne, University of Wales Institute Cardiff, UKK

EUSPRIG 2009 PROGRAMME COMMITTEE

Simon Thorne (Chair), University of Wales Institute Cardiff (UK)
David Chadwick, University of Greenwich, UK
Eric Bruillard, STEF ENS Cachan, Paris, France
Françoise Tort, STEF ENS Cachan, Paris, France
Pat Cleary, University of Wales Institute Cardiff, UK
Roland Mittermeir, University of Klagenfurt, Austria
Grenville Croll, Spreadsheet Engineering Ltd, UK
Patrick O’Beirne, Systems Modelling Ltd, Ireland
David Ward, Baker Tilly, London, UK
Ray Butler, Highways Agency, UK
# TABLE OF CONTENTS

<table>
<thead>
<tr>
<th>Author</th>
<th>Paper</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>Patrick O’Beirne</td>
<td>Checks and Controls in Spreadsheets</td>
<td>1</td>
</tr>
<tr>
<td>Susan Allen</td>
<td>Excel Modelling, Transparency, Auditing and Business Use</td>
<td>9</td>
</tr>
<tr>
<td>Tom Grossman, Ozgur Ozluk &amp; Jan Gustavson</td>
<td>The Lookup Technique to Replace Nested-If Formulas in Spreadsheet Programming</td>
<td>17</td>
</tr>
<tr>
<td>Leslie Bradley &amp; Kevin McDaid</td>
<td>Error Estimation in Large Spreadsheets using Bayesian Statistics</td>
<td>27</td>
</tr>
<tr>
<td>Angus Dunn</td>
<td>There will be a better way ...</td>
<td>39</td>
</tr>
<tr>
<td>John Hunt</td>
<td>An approach for the automated risk assessment of structural differences between spreadsheets (DiffXL)</td>
<td>47</td>
</tr>
<tr>
<td>Matthew Dinmore</td>
<td>Documenting Problem-Solving Knowledge: Proposed Annotation Design Guidelines and their Application to Spreadsheet Tools</td>
<td>57</td>
</tr>
<tr>
<td>Ruth McKeever, Kevin McDaid &amp; Brian Bishop</td>
<td>An Analysis of the Impact of Named Ranges on the Debugging Performance of Novice Users</td>
<td>69</td>
</tr>
<tr>
<td>Bill Bekenn &amp; Ray Hooper</td>
<td>Some Spreadsheet Poka-Yoke</td>
<td>83</td>
</tr>
<tr>
<td>Angela Collins</td>
<td>Embedded spreadsheet modelling</td>
<td>95</td>
</tr>
<tr>
<td>Derek Flood, Kevin Mc Daid, Fergal McCaffery.</td>
<td>NLP-SIR: A Natural Language Approach for Spreadsheet Information Retrieval</td>
<td>101</td>
</tr>
<tr>
<td>Sriram Iyengar &amp; John Shvirkely.</td>
<td>The Medical Algorithms Project</td>
<td>113</td>
</tr>
<tr>
<td>Françoise Tort, François-Marie Blondel, Éric Bruillard</td>
<td>From errors detection to behaviour observation: first results from screen capture analysis</td>
<td>119</td>
</tr>
<tr>
<td>Étienne Vandeput</td>
<td>Milestones for Teaching the Spreadsheet Program</td>
<td>133</td>
</tr>
<tr>
<td>Grenville Croll</td>
<td>Spreadsheets and the Financial Collapse</td>
<td>145</td>
</tr>
<tr>
<td>Sebastian Dewhurst</td>
<td>Reducing the Risk of Spreadsheet Usage - A Case Study</td>
<td>163</td>
</tr>
<tr>
<td>Jocelyn Paine</td>
<td>Documenting Spreadsheets with Pseudo-Code: an Exercise with Cash-Flow and Loans</td>
<td>173</td>
</tr>
</tbody>
</table>
PREFACE

Dear Colleagues,

Welcome to Paris and welcome to the 10th Annual EuSpRIG Conference. It is a considerable achievement for any organisation, especially a non-profit and voluntary organisation such as EuSpRIG, to be able to celebrate such an anniversary. I extend a special welcome to our keynote speakers Martin Erwig, University of Oregon, and Louise Pryor, Financial Reporting Council, together with delegates and authors who have travelled from outside of Europe to be with us this year.

I thank once again the members of our various committees, our reviewers and our hosts at ENS Cachan for their efforts in bringing the conference together. It is remarkable that so many of the people involved with EuSpRIG at the outset are still actively supporting us, and we welcome the new faces who are keen to progress our work. We thank our sponsors for their support and encouragement.

EuSpRIG has been going so long now that one of our founders, Pat Cleary, is retiring this September. It would thus seem timely to say a special thank you to our three founders – Pat Cleary, David Chadwick and Ray Butler, together with Ray Panko, who initially brought them together – for their contributions over the last ten years.

This year’s programme sees contributions from far and wide and evidence of an agenda expanding far outside our original “errors” domain. There is evidence that our work is being picked up in fields not previously associated with EuSpRIG. Conversely, thinking from well-established disciplines such as manufacturing engineering and behavioural analysis is being brought into our domain. We are again offering a twin tracked agenda on our first day so that delegates can choose the thread that best matches their interests.

As you can see, we have come up with a new logo, which we believe represents our conservative, respectable and forward-looking nature. This is now being fed into the design of a new website which we hope will be ready very soon. Once the new website is up and running, we will increase our marketing activities to promote future conferences.
Congratulations are due to Dr Simon Thorne for his doctoral achievement. He joins a small elite who have achieved doctoral status in spreadsheet related research work. Our academic colleagues continue to be successful in securing significant research grants to further work in our discipline. EuSpRIG itself is in the position of being able to fund some academic work, and we have recently resurrected the bursary committee to investigate this. We continue to upload our work to www.arxiv.org so that it remains as visible and as freely accessible as is possible. Attempts are being made to publish some of our work in mainstream journals to broaden our reach.

The credit crunch has made life difficult for many of our commercial supporters. It would be amusing were it not so tragic that the many of the purchasers of Enterprise Spreadsheet Management systems (for example) are, were or became recently insolvent and had to be bailed out by the authorities. We hope that the recession ends soon enough and further commercial progress can be made by our wide vendor community.

The conference this year is followed immediately by our biennial General Meeting, which you are most welcome to attend.

Again, welcome to Paris and enjoy the 10th EuSpRIG conference.

Kind Regards

Grenville Croll, EuSpRIG Chair
grenville@spreadsheetrisks.com
http://www.eusprig.org
http://groups.yahoo.com/group/eusprig
Checks and Controls in spreadsheets
Patrick O'Beirne
Systems Modelling Ltd
Tara Hill, Gorey, Co. Wexford, Ireland
Tel +353-53-942-2294 Email pob@sysmod.com

4 June 2009

Abstract

Spreadsheets that are informally created are harder to test than they should be. Simple cross-foot checks or being easily readable are modest but attainable goals for every spreadsheet developer. This paper lists some tips on building self-checking into a spreadsheet in order to provide more confidence to the reader that a spreadsheet is robust.

Keywords: spreadsheets, spreadsheet errors, quality, end-user computing, visualisation, data integrity, auditing software.

Eusprig 2009 conference
**Introduction**

CFO Magazine had a popular series of articles in 2008 on CFO peeves about sloppiness in spreadsheets presented to them:


Many of the complaints are about spreadsheets missing cross-foot checks or not being easily readable. This paper lists some tips on building self-checking into a spreadsheet in order to provide more confidence to the reader that your spreadsheet is robust. These techniques correspond to the classical programming technique of assertions and the recent trends to test-driven development.

**1) Cross foot**

The cross-foot check is to sum every column and every row and compare the sum of the grand total row with the sum of the row totals in the final column. Calculate the difference; if it is non-zero the amount may help you identify where the error is. Have a cell that displays a large red error indicator if the difference is significant, so that even on printouts it cannot be missed. Excel always has small rounding errors from floating point arithmetic, which although as small as 1E-13 (10^{-13}) will still be different from zero. Therefore you should test for the absolute value of the difference being greater than 0.01 or whatever number suits the scale of values you are working with. Here is an example of such a formula; it shows a blank if the two values are close enough, otherwise it displays a message, which for emphasis can be in formatted in a large font and red text:

\[
\text{=IF( ABS(H10-J10)<0.01, "", "Totals across and down do not match" )}
\]

A good sign of a careful spreadsheet maker is that the front sheet (or whatever sheet is always printed) carries forward the error check indicator from every supporting sheet, so that a report cannot be embarrassingly printed without knowing that some supporting sheet has an error.

We also want to avoid the risk of forgetting to include a row in the subtotal or total rows. One way to check the grand total is to sum the entire table range including intermediate totals, grand totals, and right-hand totals, and divide that by four.

**2) Balance**

In Accounting 101 you learn that a balance sheet must balance; eg by defining owner's equity as a balancing line equal to total assets less total liabilities. Chemical engineers prove their calculations by a mass balance that shows how all the material inputs emerge as outputs. To prove that your spreadsheets have some data integrity, you need to look for any opportunity to verify that a total of inputs is the same as the total of outputs.

For example, if you have a total budget to allocate, the sum of the allocated amounts must equal the original total. Percentages should add up to 100%.
3) Proportion

Look for opportunities to calculate averages, to divide total sales or labour costs by number of employees and compare with previous ratios. When forecasting projects, there is often no known input amount to compare with an output. In such cases we look at the absolute value of changes or the proportional value of changes to see if they correspond. One might compare last year's increase with this year's to see if they are as expected. Unfortunately, expectation is a dodgy measure that works in stable conditions but not in the unstable real world - see 'Expectations' below.

4) Multiple plus ungood ¹

We've all seen this kind of total calculation:

=B11+B17+B27+B37+B48+B67

and know there is a high likelihood of a pointing error - that one of those references is pointing to the wrong cell; or that there is some reference overlooked in creating that chain. It arises when there are detail lines, intermediate totals calculated by =SUM() and to get a total of these intermediate figures, each needs to be individually selected. Here are some ways to get around this:

4a) Assuming that every detail figure is also included in the =SUM() intermediate formulas, simply calculate =SUM(B2:B67)/2.

4b) Replace all the =SUM() formulas with =SUBTOTAL(9,... Then replace the multiple-plus formula with =SUBTOTAL(9,B2:B67). The SUBTOTAL function ignores the results of other SUBTOTAL functions in its range.

If the data is an export from an accounting system or database, or is a simple table of numbers, you can get Excel to automatically put in the subtotal formulas, so you don't have to. Ensure that every column has an identifying header; and that every row has an entry in a column that indicates what group this row belongs to, and the table is sorted by that column. Then choose the Data Subtotals command, specify the grouping column, and check all the data columns that you want subtotalled; and Excel puts them in for you. When you insert or delete rows in the table, simply re-do the Subtotals command.

4c) There may be other intermediate calculations that are only required for display and not for further calculation. They could be SUM or cumulative calculations that give a running accumulation from left to right. A way to exclude their result is to use the text functions =FIXED() and =DOLLAR() which produce text results that are excluded from SUM totals. Be aware that Excel will still treat them as numbers if they are referred to individually in a formula.

¹ This is a play on “doubleplusungood” which in George Orwell’s 1984 is Newspeak for “very bad”
5) **Room for expansion**

How do you make sure the totals still refer to the correct cells if rows are inserted or deleted? Take for example a formula in B67 =SUBTOTAL(9,B51:B66)

If you insert a row at row 51 the formula now reads =SUBTOTAL(9,B52:B67)
If you insert a row at row 67 the formula still reads =SUBTOTAL(9,B51:B66)

Either way, the row you inserted is excluded from the calculation. On the Tools > Options > Edit tab there is a setting "Extend data table formats and functions" that may automagically² adjust the formulas for you, but I would not rely on it.

To avoid the risk of missing an inserted row or column, always make sure every range to be summed begins and ends at a blank cell. If the first row consists only of a heading in the first column, that is readable enough. If not, you can enter in the first column the prompt "(Insert further rows below this line)" and in the last row "(Insert further rows above this line)". You could also fill the cells with ten underline characters, or make the interior colour black or blue and make the row a smaller height.

Another way to protect against insertion at the bottom of the column is: in B67 enter =SUBTOTAL(9,B51:OFFSET(B67,-1,0))

That looks like a circular reference to B67 in B67 but in fact the OFFSET function is interpreted as a reference first so in B67, OFFSET(B67,-1,0) is interpreted as B66.

http://www.mvps.org/dmcritchie/excel/offset.htm points out an alternative using INDEX to always refer to the row above:

=SUBTOTAL(9,B51:INDEX(B:B,ROW()-1))

However, both of those are more difficult for general Excel users to understand, and neither protects against insertion at the top of the range.

6) **Other sources of information**

In section (1) I gave an example of a warning message in a formula that checks cross-foot values. In programming terms this is termed an assertion. In VBA a similar concept is the Debug.Assert method which suspends execution if the test specified returns FALSE, eg

1000 Debug.Assert ThisValue=ThatValue

In a previous Eusprig paper “Information and Data Quality in Spreadsheets” [5] I listed in Appendix 2 features of Excel such as Data Validation, Conditional Formatting, and techniques for navigating, finding, and selecting data.

The Spreadsheet Safe syllabus [6] for the Spreadsheet Safe certification examination includes many of these techniques.

---

² This is a play on “automatically, magically”

Phil Bewig’s paper “How do you know your spreadsheet is right” [4] is available from the eusprig.org website.

In a paper at Eusprig 2008 [8], David Colver claimed that “Operis and smaller accountants have an integrity testing formula for every 3 to 14 formulae […], the large banks and accounting firms have such a formula for 200 to 250 formulae.” He listed fourteen tests specific to the financial domain:

1. Balance sheet balances
2. Financial statements add up
3. Financial statements have expected signs
4. Sources match uses
5. Identities hold true
6. Balance sheet clears out
7. Cash cascade gives same net cash figure as the cash flow
8. Ratio Inclusion analysis [See Colver, D, Eusprig 2007 proceedings]
9. Tax reconciliation
10. Yield analysis
11. Physical identities
12. Iterative solution has converged
13. Inputs make sense
14. Outputs meet participants’ requirements

7) Expectations

An obvious flaw in relying on expectations is getting the answer you expect instead of the correct answer. Another is getting an answer that is materially different but not obviously so. It is unavoidable that we tend to look at the new in terms of the familiar, and when that raises questions it’s useful. But when it does not raise questions, it does not mean the answer is correct - merely that that particular test gave us no information.

Remember the Dilbert cartoon where he says to a person offering a spreadsheet checking "I don't think accuracy matters if no one can tell what it's for", and that person reports back to the puzzled boss "And Dilbert found no inaccuracies!" Some people are better than others at finding errors, just as good developers can be many times more productive than bad ones.
8) The Top Ten spreadsheet questions

In order to achieve quality and robustness in spreadsheet use, spreadsheet owners should provide answers to the following questions.

1. **What is the purpose of the spreadsheet?**
   a. Criticality – what if it were lost or corrupted?

2. **Where is it kept – network location, set of files**
   a. How do we know which is the current version?
   b. Complete list of data sources it depends upon
   c. What depends on this spreadsheet?

3. **How is it used? (Process documentation, instructions)**

4. **Is it for one person or is it re-used by others?**

5. **Is it once-off (project) or has it a periodic operation?**

6. **Who peer reviews its structure and version changes?**
   a. If none, likelihood of key-person risk?
   b. Evidence of test (with results) and acceptance

7. **What controls are around it?**
   a. Who reviews & signs off its output?
   b. Reconciliation with data sources

8. **What checks are included within it?**
   a. Cross-foot, balance checks, etc

9. **What evidence is there of conformity to good design practices?**
   a. Potential long list, see “Spreadsheet Check and Control” book
   b. Clear block layout, formats, print output header/footer
   c. Formula integrity, protection, no errors, no external links
   d. Use of timesaving formulas and features

10. **What are the pain points?**
    a. Quality of input data; duplication, update
    b. Grunt work transforming data
    c. Effort maintaining & updating formulas
    d. Training in more efficient Excel skills
    e. Possible to replace with controlled shared system?
References


© Patrick O’Beirne June 2009
www.sysmod.com
Excel Modelling - Transparency, Auditing and Business Use

Susan Allan, Risk Analytics, Lloyds Banking Group
susan_allan@bankofscotland.co.uk

ABSTRACT

Within Lloyds Banking Group the heritage HBOS Corporate division deals with Corporate loans, and is required to assess these loans for risk in accordance with the Basle Accord regulations. Statistical Risk Rating models are developed by the risk analysts to assess the obligors credit worthiness. It is necessary then to provide the bankers who originated the loan (‘Relationship Managers’ or RMs) with an assessment tool to generate the loan rating upon which they base their lending decisions. Heritage HBoS Corporate required a new model build system for holding its Risk Rating models in 2006 as a result of more complex models being created to comply with the Basle Accord. The use of Excel was promoted by the IT department for a number of reasons; the Excel solution now in use is reviewed in this paper.

1. INTRODUCTION

This paper discusses the benefits and considerations employed in the use of the new Excel-based model build system, looking first at the issues with visibility to users, then at the good-practice guides that have been used within the company.

The paper then looks at issues that have arisen through the development of the new system, and how processes have been reviewed as a result of its use.

2. BACKGROUND

When the RM has input the deal information in the front-end system ('Nexus') and generated a rating, this information then feeds into the models in the new model build system ('Nexus Model Designer' or NMD) to produce a rating for that loan, which feeds back into Nexus to be viewed by the RM. The loan is then passed via Nexus to the Sanctioners for review and approval, before the customer is advised of the loan price. The RMs and Sanctioners expressed a need to view the Risk Rating model when specific deals are entered so that they can query how a deal is being rated in a timely manner thus allowing a quick response to the customer.

3. TRANSPARENCY

In previous model build systems there had been no visibility of the risk models to anyone but the analysts permitted access to them. This meant that as models were developed there was no way of showing the RMs what they looked like without providing them with access to the system and subsequent training in it; given that the locations of some of these managers was spread around the country this was not practical.

It had become clear to the Risk Analytics model implementation team that there were a number of gaps in the current processes; as well as the ability to extend the visibility of
models to the RMs throughout the model development, a full model audit trail was required by the internal and external auditors.

To this end the NMD Excel-based tool seemed to fit the requirements of being accessible to all, and using an existing system (Excel) which would require minimal training; the view was that most users had basic to moderate understanding of Excel functions.

4. NMD: GOOD-PRACTICE GUIDELINES EMPLOYED

Given the time already spent by the RMs in their 'day jobs' it was important that the users could work out in a short time period how their deals flow through the model and what the model is doing with each calculation. To this end good practice [Grossman & Ozluk 2003][Cernauskas, Kumiega, Van Vliet, 2007] [Bishop, 2006] meant using the following steps:

4.1 Use a structured spreadsheet

Best practice within HBoS follows the guidelines of having a structure which uses separate input, output and calculation sheets. This enables identification of changes to inputs/outputs and calculations, an important point for the system development to meet auditing requirements which are discussed in section 4.3. Within the NMD there is a 'formula walker' which enables the user to step through the formula from start to finish, and records the steps taken so that these can be passed to RMs/sanctioners to demonstrate how a deal gets its rating in a particular model.

The formula walker, shown below in Figure 1, has significantly improved the existing “Trace Precedents and Dependents” functionality by showing the sheet and cell name, formulae and values in one place.

Figure 1 – the NMD Formula Walker

4.2 Ensure the formulae use common easily-interpreted functions.

Where complex formulae exist it may be worth considering a 'wizard' to better detail what the formula is doing.

For example, in the HBOS Corporate models complex array formulae are used to conditionally map values across spreadsheets, and these are often difficult to interpret; more so if the conditional formula is extended with several conditions. To aid with this, a wizard was created in the NMD which builds such formula through a graphical interface; an example is shown in Figures 2a and 2b.
4.3 Auditability

In the NMD each time a change is made to the model it is logged. This ensures that the internal and external audit requirement around transparency of change is covered. To ensure that the regulatory requirements [Financial Services Authority, 2006] are met around the use of the rating system throughout the company and in the decisioning process, input is required from a number of parties (such as RMs, IT teams, and data teams who may need outputs named in a structured manner for the databases).

The NMD was built to log two types of changes – one where the internal model formula/lookups change, and the second where the inputs or outputs change. For the first, a new ‘revision’ is logged when the amended model is saved. For the second, a new version number is created when it is saved. This new version number is referenced by the IT department when the model is promoted into the live environment for use with live deals.

With each change logged, the user name would be logged along with the time and date of the change in the NMD (Fig 3a). It is then visible when the change was made, what the change entailed and who had made the change.

This is a step forward from the lack of transparency in previous systems, and is an important step in terms of ensuring the analysts can trace back to see why particular changes were entered. With each change a description can be entered. Given the number of iterations of models, versions can also be archived where appropriate within the NMD for recall at any time.

\begin{verbatim}
=MAX (SecDI.ExposureResidualMaturity
[SecDI.SecurityID = SEC_GteeADJ.SecurityID AND
SecDI1.LinkFlag = 1 ] )
\end{verbatim}

The condition is shown here in the square brackets. This says that this cell will take the maximum of the exposure Residual Maturity value from the column with these values in the SecDI sheet, if the security IDs for this security on this sheet (the SecGteeADJ sheet) and row match the security ID on the same row in the SecDI sheet, and also where the link flag on the SecDI sheet = 1 on the same row within the LinkFlag column.

\begin{verbatim}
=MAX(IF(SecDI!$B$5:$B$754=SEC_GTEEADJ!$B5,
IF(SecDI1!$C$5:$C$754=1,SecDI!$L$5:$L$754)))
\end{verbatim}

Same thing being said here but as an array formula – no names for the ranges, nor for the cells so interpretation is more difficult. Naming on the columns/cells in the box above is made possible using Excel’s named range function.
5. USER FEEDBACK AND LESSONS LEARNT

When the models are exported to the RMs, the assumption was that there would be sufficient knowledge of Excel to enable interrogation of the models as they are developed. Whilst functions such as the conditional mapping formulae can be explained, and were easily picked up by the RMs, there was some surprising feedback around the look and feel of the Excel view presented by the NMD models.
In previous simple models which had been replicated from the old model build system into Excel spreadsheets, much use had been made of Visual Basic (VB) coding to make them look glossy and easy to use for the RMs as shown in figure 4 above. The expectation from the RMs was that models from the NMD would have the same look as the simpler VB models they had seen previously. These required minimal interpretation and the expectation was that as both were Excel models there should be little difference. However with the NMD models, VB cannot be used as it is deemed by the IT designers as a corruptible program which cannot be controlled. Furthermore Microsoft will not support VB going forwards as there is a move to use .NET instead. As a result the look of models exported to the RMs from the NMD is simply that of a basic Excel spreadsheet with minimal formatting as depicted in Figure 5.
In addition the input sheets in the models follow the object structure observed in the Nexus database rather than the financial statement layout, again something with which the RMs are not immediately familiar.

The models can now be exported to all users, which is a major step forwards. However the planning around how the RMs and sanctioners would use this new toolset has been underestimated and needs further attention to ensure model build projects are a complete success. As a result of the feedback, the model developers have now instigated a phase in the model build projects where they will discuss the models with the users and discuss with them how the structures are presented. This alleviates any potential confusion around data structures and the look and feel of the models.

The NMD project was driven by the Risk Analytics Implementation team requirements and has satisfied many requirements in terms of building a tool which could interpret deals end to end, as well as having added further functionality around auditing. It is interesting to see that the people aspect [Grossman, 2002], whilst it may be frequently missed, is seen time and again across organisations. David Elton, in his article in the Financial Times [Elton, 2008], notes that IT is all about people and their reactions and feelings. He notes that “changing the way people work, not just the technology, has to be at the core” (of every IT project).

This is where the challenge now lies. The technology is now readily available, it is now necessary to ensure that all parties can see the benefits and adapt to looking at Excel in a slightly different way to their usual VB-adapted screens.

6. CONCLUSIONS

The NMD project has addressed most concerns raised by the various model users. The interface shown to the RMs is a minor issue and therefore an aspiration to look towards. Time spent up-front in model build projects to engage the RMs and explain the look of
the NMD models will avoid time spent post model build convincing users of the benefits of using Excel in the NMD.

These benefits include ease of access to a system available to all to view the models, which is a major step forwards from previous model build systems which were viewed as 'black boxes'. There is now full transparency for the RMs through the model calculations using either simple Excel functions or in the more complex cases by using a wizard function i.e. for arrays.

In addition the system has been built to include a version control function which picks up all changes to Excel sheets, and differentiates between model inputs/outputs which will affect the model structure, and the internal calculations. This change control function provides full confidence in the auditing of all changes.

7. REFERENCES


The Lookup Technique to Replace Nested-IF Formulas in Spreadsheet Programming

Thomas A. Grossman
University of San Francisco, School of Business & Management, San Francisco CA 94117-1045
tagrossman@usfca.edu

Özgür Özlük
San Francisco State University, College of Business, San Francisco, CA 94132
ozgur@sfsu.edu

Jan Gustavson
Senior Financial Analyst, Visa, Inc., San Francisco, CA 94128-8999

ABSTRACT

Spreadsheet programmers often implement contingent logic using a nested-IF formula even though this technique is difficult to test and audit and is believed to be risky. We interpret the programming of contingent logic in spreadsheets in the context of traditional computer programming. We investigate the “lookup technique” as an alternative to nested-IF formulas, describe its benefits for testing and auditing, and define its limitations. The lookup technique employs four distinct principles: 1) make logical tests visible; 2) make outcomes visible; 3) make logical structure visible; and 4) replace a multi-function nested-IF formula with a single-function lookup formula. It can be used only for certain simple contingent logic. We describe how the principles can be applied in more complex situations, and suggest avenues for further research.

1. INTRODUCTION

Spreadsheets are widely used in business for a variety of important purposes (Coles and Rowley 1996, Ragsdale 2001, Croll 2005, Grossman, Mehrotra and Özlük 2007). A spreadsheet containing cell formulas is a computer program, and all computer programs are susceptible to errors. EuSpRIG maintains a corpus of spreadsheet errors (EuSpRIG 2009). Several scoping-scale research projects (summarized in Panko 2000a, 2000b; Panko 2008) suggest that spreadsheets are particularly vulnerable to errors. However, follow-on research (Powell, Baker and Lawson 2008a, 2008b, and 2008c) indicates that the prevalence and risk of errors in important spreadsheets is not well understood and further research is required.

Spreadsheet programmers often implement contingent logic using the nested-IF formula even though this technique is difficult to test and audit and is believed to be risky. In this paper, we investigate the “lookup technique” for programming contingent logic that can reduce or eliminate the need to use the nested-IF formulas. In certain circumstances, a complex nested-IF can be replaced in a mechanical fashion (without exercise of judgment nor loss of functionality) by a lookup function that is safer to program and provides other benefits.

1.1. Nested-IF Formulas Are Considered Risky

Computer programmers often need to program contingent logic. Contingent logic in spreadsheets seems typically to be programmed using a nested-IF formula. We have encountered Nested-If formulas in research observations and informal discussions with experienced spreadsheet programmers. We put out a call to the EuSpRIG list and received many real-world examples of nested-IF formulas.

A nested-IF is created when an IF function used as an argument (“nested”) of one or more other IF functions. There is much concern about the use of nested-IF formulas in spreadsheets. Lanza (2006) uses the number of nested-IF formulas as a determinant in the likelihood of risk in a spreadsheet. Croll and Butler (2006) examine the common sources of errors in medical spreadsheet applications and they find the complex nested-IF formulas as one of the main culprits. Thiriez (2004) provides a nested-IF he used that he believes is difficult to audit. Baxter (2005) discusses several spreadsheet auditing tools have been created to address the...
potential flaws in logic creation which perform checks for logic elements that are known to be particularly
error prone (e.g. nested-IF formulas). Spreadsheet audit software such as Spreadsheet Professional and
XLAnalyst flag nested-IF formulas as a source of risk.

Although nested-IF formulas may be risky, there is reason to believe that skilled software engineers can
manage the risk. As with any challenging programming task, a diligent programmer can write a complex
nested-IF accurately. For example, Grossman, Mehrotra and Özlük (2007) found an example of a nested-IF
formula with 5 levels of nesting and 11 IF functions that are believed to be correct. However, the prevalence
of such diligent programming practices is unknown.

1.2. Contribution

This paper considers the “lookup technique” that a programmer in certain circumstances can use instead of a
nested-IF formula. The technique uses a lookup function (specifically VLOOKUP or HLOOKUP) to obtain
the same functionality as a nested-IF. The general approach has been known for some time. We hypothesize
that many spreadsheet programmers have independently discovered it over the years. Read and Batson (2000
p. 6-55) discuss this concept in passing. Jelen (2008) recommends it and focuses on the syntax of the
VLOOKUP function.

The contribution of this paper is to present the principles underlying the lookup technique, describe its
benefits for testing and auditing, define its limitations, describe how the principles can be applied in more
complex situations, and suggest further research.

We interpret the programming of contingent logic in spreadsheets in the context of traditional computer
programming. We discuss the problems of nested-IF formulas for spreadsheet testing and spreadsheet
 auditing. We explain why the lookup approach is beneficial for testing and auditing.

We decompose the lookup method for programming contingent logic into its component parts. We conclude
that the lookup approach consists of the application of four distinct principles: 1) make logical tests visible; 2)
make outcomes visible; 3) make logical structure visible; and 4) replace a multi-function nested-IF formula
with a single-function lookup formula.

We show the limitations of the lookup approach, which is applicable only for nested-IF formulas with certain
well-defined characteristics. For those nested-IF formulas that do not admit the use of a lookup function, we
believe there are benefits from providing visibility into the logical tests, outcomes, and logical structure. We
discuss areas that merit further research.

1.3. Structure of Paper

In section 2 we discuss contingent logic in traditional computer programming and point out that nested-IF
formulas were once the norm. We describe nested-IF formulas in spreadsheets and discuss the problems posed
for testing and auditing. In section 3 we develop an alternative called the “lookup technique”, and show how
this technique employs four distinct principles. In section 4 we evaluate the lookup technique. We compare it
with spreadsheet engineering recommendations, consider resource consumption, and explain when it can and
cannot be applied, and discuss the principles of visibility applied to more complex situations, and the
challenge of “acceptable errors”. We conclude in section 5.

2. NESTED-IF FOR SPREADSHEET CONTINGENT LOGIC

In this section we discuss contingent logic, including nested-IF in traditional programming languages and in
spreadsheets, and describe the difficulties of testing and auditing a spreadsheet that contains nested-IF
formulas.

2.1. Contingent Logic in Computer Programming

This paper addresses an important aspect of spreadsheet computer programming, namely the implementation
of contingent logic. Contingent logic is required when the programmer wants a computer program to return
different values depending on the state of one or more other values.

Traditional programming languages handle contingent logic using programming statements such as If-Then-
Else, If-Then-ElseIf, Case, and Switch. Early scientific languages such as FORTRAN 66 provided only the
If-Then-Else statement which required cumbersome nesting. Note that the challenge posed by nested-IF predates the spreadsheet!

Later programming languages, such as FORTRAN 77, reduced the need for nested-IF constructs by providing the more powerful If-Then-ElseIf statement. Some modern programming languages also provide the flexible Case and Switch statements.

To better program complex logic, traditional programmers devised conventions such as indenting, line breaks, and color coding. These conventions are universal in traditional programming and are often automated (for example, the Visual Basic editor). However, there are no such standard conventions for programming complex logic in spreadsheets. Indeed, Panko (2008) states that spreadsheet programming “seems to resemble programming practice in the 1950s and 1960s”, suggesting there is significant opportunity for improvement in productivity, maintainability, and accuracy. Just as the advent of the If-Then-ElseIf statement took traditional programming out of the 1960’s approach of using nested-If statements, we hope that the lookup technique and follow-on research can advance the practice of spreadsheet programming.

2.2. Nested-IF in a Spreadsheet

Contingent logic is required when a cell in the program must return different values depending on the value of an input or an intermediate calculation. Contingent logic in spreadsheets is typically programmed using the IF function. The IF function allows the programmer to implement contingent logic that returns one outcome (the “value_if_true” outcome) if the logical test evaluates TRUE, and another outcome (the “value_if_false” outcome) if the logical test evaluates to FALSE. The IF function syntax is as follows:

=IF(logical_test , value_if_true , value_if_false)

When more than one contingency is present, the programmer can nest one IF function inside another. Any of the arguments of the IF function can themselves be functions, providing a high degree of flexibility. For example,


2.3. Difficulty Testing and Auditing Nested-IF

The simple example above is easy to understand. There are but two IF functions, and the outcomes (value_if_true and value_if_false) are text. However, the value_if_true and value_if_false can themselves be IF functions. This nesting process can lead to cell formulas of substantial complexity. Consider this nested-IF formula, drawn from a regularly used spreadsheet1:

=IF($D14="Rev",$C14*E$11/$C$11,IF($D14="Units",$C14*E$9/$C$9,IF($D14="MH",$C14*E$12/$C$12,IF($D14="MC",$C14*E$13/$C$13,IF($C14="","not correct Allocation"))))).

In order to better visualize the formula, we rearrange it and strip out the “$” absolute reference markers:

=IF(B10="Rev",B14*B15/B16,IF(B10="Units",B14*B17/B18,IF(B10="MH",B14*B19/B20,IF(B10="MC",B14*B21/B22,IF(B11="","not correct Alloc."))))).

We program this nested-If in Figure 1.

---

1 This nested-IF formula was provided by an informant who indicated the spreadsheet is in routine use.
This nested-IF formula poses challenges for a programmer who wants to verify accuracy. One approach to verifying accuracy is to audit the cell formula, by having a programmer (or team of programmers, Panko 2008) inspect the formula. However, it is not a simple exercise to understand the purpose of such a formula. Even after the purpose is understood, it is difficult to verify its correctness.

There are three challenges in auditing this formula. First is the accuracy of the logical test formulas. Second is the accuracy of the outcome formulas themselves; there are six different outcome values in this formula. Each must be separately verified. Third is the accuracy of the contingent logic structure, which is controlled by the location of the five IF functions and the match between the logical tests and the outcomes.

Another approach to verifying accuracy is to use traditional software testing. To test a spreadsheet containing a nested-IF formula, it is necessary to devise test cases that will toggle every possible state of the nested-IF. Such test cases can be challenging or even impossible to discover. Therefore, traditional input-output testing is challenging and might be impossible.

3. THE FOUR PRINCIPLES OF THE LOOKUP TECHNIQUE

To ameliorate these problems, programmers sometimes use the lookup technique, which replaces the nested-IF formula with several additional cells and a simple lookup function, as shown in Figure 5 below.

The lookup technique employs four principles. First is to make visible the logical test values (TRUE or FALSE). Second is to make visible the outcome values. Third is to make visible the structure of the contingent logic. Fourth is to use a simple formula to handle the contingent logic. To elucidate the principles underlying the lookup technique, we will take a nested-IF and transform it step-by-step into a lookup.

3.1. First Principle: Make Visible the Logical Test Values

We rewrite our example nested-IF to make visible in the spreadsheet the values of the logical_tests. We replace each logical_test formula with the range name “Test1”, “Test2”, etc. This yields the following nested-IF formula.

\[
=\text{IF}(\text{Test1}, \text{SBS14}\times\text{SB15}/\text{SB16}, \text{IF}(\text{Test2}, \text{SBS14}\times\text{SB17}/\text{SB18}, \text{IF}(\text{Test3}, \text{SBS14}\times\text{SB19}/\text{SB20}, \\
\text{IF}(\text{Test4}, \text{SBS14}\times\text{SB21}/\text{SB22}, \text{IF}(\text{Test5}, \text{""}, \text{"Not Correct Alloc."}))))))
\]

where

- Test1 replaces SBS10="Rev"
- Test2 replaces SBS10="Units"
- Test3 replaces SBS10="MH"
- Test4 replaces SBS10="MC"
- Test5 replaces SBS11=""
We can program this in a spreadsheet as shown in Figure 2.

![Figure 2: Logical Test Values Made Visible in cells E10:E14](image)

Notice that we added five new formula cells, one for each of the five logical tests. This allows each logical test to be instantly verified for accuracy, whatever the state of the spreadsheet.

3.2. Second Principle: Make Visible the Outcome Values

We next rewrite our example nested-IF to make visible in the spreadsheet the values of the Outcomes. We replace each outcome (formula or text) with the range names “Value1”, “Value2” and so forth. This yields the following nested-IF formula.

$$\text{=IF(Test1, Value1, IF(Test2, Value2, IF(Test3, Value3, IF(Test4, Value4, IF(Test5, Value5, Value6))))}$$

where Value1 through Value6 is defined as above and

- Value1 replaces $B$14*$B$15/$B$16
- Value2 replaces $B$14*$B$17/$B$18
- Value3 replaces $B$14*$B$19/$B$20
- Value4 replaces $B$14*$B$21/$B$22
- Value5 replaces 
- Value6 replaces "not correct Allocation"

We can program this in a spreadsheet as shown in Figure 3:
Notice that we added six new formulas cells, one for each of the six outcomes. This allows each outcome to be instantly verified for accuracy, whatever the state of the spreadsheet. If we intend to perform traditional testing, these outcome values can be tested for every test case, not just for the test cases that cause the contingent logic to display that outcome.

3.3. **Third Principle: Make Visible the Structure of the Contingent Logic**

Our next improvement is to modify the spreadsheet to make the structure of the contingent logic more visible. We rearrange the cells so that the logical tests are placed next to their corresponding outcome values.

Notice in Figure 4 that if Test1 is TRUE, then the value in the same row is returned. If Test1 is FALSE and Test2 is TRUE, the value in the same row as Test2 is returned, and so forth. If all five logical tests are FALSE, the final “(otherwise)” value is returned.
3.4. Fourth Principle: Simplify the Formula by Replacing Nested-IF with a Lookup

Our next improvement is to replace the complex, multi-function nested-IF cell formula with a simple, single-function VLOOKUP cell formula. The new formula is

\[ \text{VLOOKUP}(\text{TRUE}, E12:F17, 2, \text{FALSE}) \]

Notice that we must type the logical test value TRUE in cell E17 to cause the VLOOKUP to return the last Outcome value. Figure 5 presents a spreadsheet with the VLOOKUP formula and the equivalent nested-IF formula.

Compared to Figure 4, Figure 5 replaces the complex nested-IF formula with a simpler VLOOKUP that is much easier to verify.

Compared to Figure 1, in Figure 5 the single nested-IF cell has been replaced by 12 cells. Five cells report the value of the five logical tests. Six cells report the value of the six outcomes. One cell contains the structure (and only the structure) of the contingent logic.

4. EVALUATION AND FURTHER RESEARCH

The lookup technique for contingent logic takes a complex nested-IF that is difficult to understand, test, and audit and replaces it with new cells that are easy to understand, are testable, and are easier to audit.

4.1. Comparison with Spreadsheet Engineering Recommendations

The lookup technique for contingent logic expands a single cell with a complex formula into many cells with simple formulas. It is consistent with the programming advice to “write for the reader”. This is well-known in the traditional programming world, and is recommended by Powell and Baker textbook (2007b, page 101). In contrast, Raffensperger (2003) recommends using the fewest cells possible. This recommendation is driven by the desire to present compact reports; this is easily handled by placing reports on a separate worksheet that references the computational results.

4.2. Resource Consumption: Cells and Computation Time

The lookup technique for contingent logic increases the number of cells required by moving “covert computations” out of the nested-IF and into their own cells. If the number of cells available is a constraint, then this is problematic. For example, Thiriez (2004) describes a model that uses all 32,000 rows of an Excel 2003 spreadsheet. With the availability of one million rows in Excel 2007, the availability of cells should be an issue rarely if ever.
The lookup technique could increase computation time because evaluations of logical tests and values that were previously computed only when needed by the contingent logic are now always evaluated. However, with modern computer hardware this issue will likely only rarely be a problem.

4.3. **Branch-on-False**

The nested-IF formulas in this paper have a pattern to the structure of the logic that we call “branch-on-False”. In the upper-level IF functions the value\_if\_true is an outcome and the value\_if\_false is a nested-IF:

\[
\text{IF(logical\_test , value , IF(…))}.
\]

The lookup technique we describe works directly only for branch-on-false. This limitation should be explored by further research. First, with simple modification to our approach it should be easy to make the lookup technique work for “branch-on-true”, which is of the form \(\text{IF(logical\_test , IF(…) , value)}\). Second, by selectively reversing the logical tests it should be possible to convert a nested-IF that has a mix of branch-on-true and branch-on-false into a nested-IF that has only branch-on-false, thereby admitting the use of the lookup technique.

4.4. **Complexity of Logical Branching**

The act of making the structure of the logic visible is difficult in the situation where there is an IF function that has further IF functions for both value argument, which has the following form:

\[
\text{IF(logical\_test , IF(…) , IF(…))}.
\]

It is not possible to directly apply the lookup technique on this type of complex contingent logic. Therefore, such nested-IFs do not admit the lookup approach. This topic merits further research; what can we do to enhance the programming of complex contingent logic?

4.5. **Principles of Visibility Applied to Complex Branching**

With complex branching, it is not possible to replace a multi-function nested-IF formula with a single-function lookup formula. Whether there is an option to make an improvement merits further research. Regardless, the “visibility” benefits seem to be available. These include making logical tests visible; making outcomes visible; making logical structure visible. Further research should explore how to apply these three principles to complex branching.

4.6. **Acceptable Errors**

In some nested-IF formulas we have seen, outcome values can evaluate to an error when the outcome value is not returned by the nested-IF. An example is a divide-by-zero error that is trapped by the nested-IF. If the programmer applies the principle of making outcome values visible, such error values will now appear in the spreadsheet. If it is unacceptable to have error values in the spreadsheet, then the error message might need to be detected and flagged in some fashion. This topic merits further research.

5. **CONCLUSION AND FUTURE DIRECTIONS**

We discuss the “lookup technique” for programming contingent logic in spreadsheets. We illustrate a step-by-step process to convert a simple nested-IF into a lookup. We explain the benefits in terms of the lookup technique in terms of testing and auditing.

We identify four principles associated with the technique. We propose that these four principles can and should be used during spreadsheet design. If they are implemented during design and programming of a spreadsheet, we hypothesize that they will reduce the total effort to create the spreadsheet (especially when one considers how long it takes to code a complex nested-IF formula), and reduce the time to debug the spreadsheet. Such hypotheses merit empirical evaluation, and we believe deployment of this approach to programmers to be an important avenue of future research.

We show that the lookup technique does not apply to certain “complex” contingent logic situations that present as a complex nested-IF formula. It would be desirable to determine what particular types of spreadsheets benefit from the proposed approach and what types may not. We anticipate that some of the principles can still be applied and yield benefits. We illustrated the lookup technique in a “spreadsheet remediation” situation, by converting an existing nested-IF into a better lookup technique.
The lookup technique yields code that in our opinion is more readable because individual cell formulas are shorter. We further believe that the task of verifying or auditing the accuracy of contingent logic is easier with the lookup approach than with the nested-IF approach. These assertions merit empirical testing.

As with any programming principles, there will always be times when a programmer chooses not to adhere to them. The four principles and the use of the lookup technique in general, should be viewed as guidelines rather than rigid rules.

We view the use of the lookup technique, and any future more advanced techniques for programming contingent logic in spreadsheets to be comparable to the traditional programming norm of systematic indentation of a complex series of statements. It is interesting to consider whether other such spreadsheet norms could be developed. Similar to traditional programming languages, we would like to see research to develop the spreadsheet equivalent of a Case statement.

Finally, a referee drew our attention to the EUSES corpus as a repository of spreadsheets. We hope to use this corpus in future research efforts.

ACKNOWLEDGEMENTS

We would like to thank the EuSpRIG program chair and the anonymous referees for their constructive feedback. Any errors of omission or commission are the responsibility of the authors.

REFERENCES


---

2 http://esquared.unl.edu/wikka.php?wakka=EUSESSpreadsheetCorpus


Error Estimation in Large Spreadsheets using Bayesian Statistics

Leslie Bradley, Kevin McDaid
Dundalk Institute of Technology, Dundalk, Ireland
leslie.bradley@dkit.ie, kevin.mcdaid@dkit.ie

ABSTRACT

Spreadsheets are ubiquitous in business with the financial sector particularly heavily reliant on the technology. It is known that the level of spreadsheet error can be high and that it is often necessary to review spreadsheets based on a structured methodology which includes a cell by cell examination of the spreadsheet. This paper outlines the early research that has been carried out into the use of Bayesian Statistical methods to estimate the level of error in large spreadsheets during cell by cell examination based on expert knowledge and partial spreadsheet test data. The estimate can aid in the decision as to the quality of the spreadsheet and the necessity to conduct further testing or not.

1. INTRODUCTION

Spreadsheets today are used in many industries throughout the world for many different purposes such as decision making, budget forecasting, corporate expansion, and risk analysis to name a few. Spreadsheet technology is a fundamental reporting and decision making tool in most sectors of business. Some areas like the financial sector employ small armies of people who use spreadsheets on a daily basis. In the city of London, [Croll, 2005], one interviewee stated in reference to the finance sector that,

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

The Sarbanes-Oxley Act of 2002 has forced companies to examine the role spreadsheets play in their financial reporting and decision-making processes. This can require an increase in company resources to adhere to such regulations. There have been many real world examples of spreadsheet errors, loan amounts in a spreadsheet from the Agricultural Bank of Namibia varied from N$59.5 million in one spreadsheet to N$50.4 million in another, [EuSpRiG, 2008].

Spreadsheets continue to be used even though spreadsheet developers use little or no standard development practices from the software industry. This has given rise to a high level of error in spreadsheets. Research into the impact of errors, [Powell, 2007b], discovered that the impact of errors varies depending on the organisation. For 25 operational spreadsheets, ten spreadsheets showed an error impact ranging from $216,806 to $110,543,305; six spreadsheets had errors with no impact while nine had no errors.

An important element of spreadsheet development is testing, where spreadsheets are examined by professionals to ensure their correctness. In the city of London, [Croll, 2005], reports that financial modelling was being limited by the 256 column limit in Microsoft Excel. The author also reports that spreadsheets greater than 1GB in size are
already in existence. Thus, to examine each cell can be very time consuming due to their size.

This paper investigates how the overall error rate of a spreadsheet can be inferred from a combination of expert knowledge or belief and partial test result information. This information can be used to decide if further cell by cell testing is required or even whether the spreadsheet has such a high level of error that it should be redeveloped. The methods are ideally suited to large spreadsheets.

The layout of the paper is as follows. Section 2 details aspects of the research into spreadsheet error. In Section 3 the paper describes the problem the research is attempting to address. The research questions to address this problem are presented in Section 4. The statistical model for error rate estimation is described in Section 5 and Section 6 explains how the model is applied to real spreadsheets with a focus on suitable measures for decision making.

2. SPREADSHEET ERROR RESEARCH

Reduction of errors in spreadsheets has been the focus for many people in the research community. There have been various investigations into possible influences of error in spreadsheets.

2.1 Human Impact

One approach taken is to study the human impact on spreadsheets. Baker, et al [Baker, 2006], conducted a questionnaire online to investigate the developer’s competence in creating/editing spreadsheets. They discovered that spreadsheet practices differ greatly from person to person and organisation to organisation. In [Benham, 2005], the authors attempt to reduce the number of errors in spreadsheets by introducing a training technique. They believe that if users have more “error awareness” that they are less likely to make errors.

Bishop and McDaid [Bishop, 2008] looked at the possibility that the level of experience in spreadsheet design can influence the user’s performance and behaviour in debugging spreadsheets. In [Panko, 1996, 1998], the author suggests that when humans do simple tasks like typing they can make undetected errors in 0.5% of all actions. This rises to 5% when doing more complex activities like writing a program or a paragraph of text. Thorne and Ball, [Thorne, 2005] look at human factors and their impact in spreadsheet development.

2.2 Spreadsheet Reviewing

Another approach used by researchers is to develop a spreadsheet reviewing methodology to detect errors. This method can often involve checking the spreadsheet manually, with the use of auditing tools or a combination of both. Powell, [Powell, 2008] developed an audit protocol that combined different techniques such as mapping and formula checking, with two auditing tools, Xlanalyst and Spreadsheet Professional.

[Panko, 2006], suggests that numerous testing practices could be used on spreadsheets by adopting software checking techniques. Panko suggests that the most extensive way of detecting errors is to conduct a cell by cell search of the spreadsheet, [Panko, 2000]. Croll, [Croll, 2003], reports on the common reviewing process for spreadsheets in the
City of London. This entails different levels of reviewing combined with the documentation of each stage.

The HM Revenue & Customs use the “SpACE” audit methodology to check spreadsheets, [HM Revenue, 2007]. The methodology contains 5 steps to audit the spreadsheet including an early decision on whether the impact of the spreadsheet justifies a complete inspection. In step three, a decision on whether the testing should begin is made by the auditor depending on factors outside of the spreadsheet such as the developer, organisation policy, problem complexity, etc.

2.3 Error Rates

Various researchers have investigated the error rates for spreadsheets. Panko, [Panko, 2005], looked at 13 studies involving operational spreadsheets that were conducted over a period of nine years, 1995 – 2004. He found an average cell error rate (CER) of 5.2% for 43 of the spreadsheets reviewed in the studies. The CER is the percentage of cells that contain errors. The results show a spreadsheet error rate (SER) of 94% for the 88 spreadsheets studied, i.e. 94% of the 88 spreadsheets contained at least one error.

A later study, [Powell, 2007a], examined 50 operational spreadsheets from various sources, such as a bank, a college and two consulting companies to name a few. The authors measured the errors in terms of error cells and instances. An instance is the occurrence of error in the spreadsheet. An instance usually involves more than one cell with the results showing an instance on average involving 10.05 cells. They discovered a much lower CER of 1.79% over 270,722 formulas.

The authors split the results into two categories; wrong result and poor practice. The wrong result section represents the quantitative errors while poor practice represents qualitative errors. For example, suppose a formula contains a hard coded number. If the result is incorrect, the cell would be deemed a wrong result. If the result is correct then the cell is judged to be a poor practice error. The CER reduced to 0.87% for wrong results, which is significant but lower than previously thought and much lower than 5.2%. Nevertheless, there is a well-accepted issue with the level of error in spreadsheets and Panko [Panko, 2007] summarises it well when he states that “the issue is not whether there is an error but how many errors there are and how serious they are”.

2.4 Issues with the Error Rates

The cell error results in [Powell, 2007a] are based on formula cells only. The results in [Panko, 2005] are based on 13 studies with the individual CER’s based on different measures. One such CER is based on high risk formulas only which do not give an accurate CER for the overall spreadsheet.

2.5 Rationale

The goal of spreadsheet reviewing is to establish its correctness. This can be almost impossible for large spreadsheets with millions of cells. Croll has reported on the size of spreadsheets in the City of London but the number of unique formulas can range from 1,000 to 10,000 and upwards, [Croll, 2003]. Instead we propose a method whereby expert information or belief as to the likely cell error rate can be combined with partial test information to infer an error rate for the remaining untested cells.
Powell, [Powell, 2007b], found 9 out of 25 spreadsheets contained no errors. Five of these error free spreadsheets came from the same organisation. Therefore it is reasonable to infer a belief that a further spreadsheet provided by this organisation will have a lower than average cell error rate. Similarly, knowledge of the developer’s ability and the spreadsheet’s complexity will influence to some extent the reviewer’s belief as to the likely cell error rate. This is important information that can be used to hone any estimate of the cell error rate. Similarly, judgements of likely quality based on the presence or absence of good practices and controls such as documentation, version control and design patterns amongst others can influence an expert’s judgement as to the likely error rate.

We feel strongly that this belief should be incorporated into the prediction model, and combined with available test information. The natural approach is a Bayesian one which we introduce in the following subsection.

2.5 Software Reliability

Research into software reliability has looked at the application of Bayesian networks to predict software reliability with limited or uncertain data. The use of Bayesian statistics allows the combination of different types of information available to the developer, [Fenton, 2004, 2007b]. The use of Bayesian networks can include qualitative information e.g. staff experience and testing quality, [Cockram, 2001; Fenton, 2007b; 2007a].

Researchers using Bayesian methods to predict software reliability, [Fenton, 2007b] suggest “that a model constructed using expert judgement and historical data, within one organisation can be used within the same organisation to predict accurately the outcome of new projects.” This research is looking at the use of Bayesian methods to combine expert information with test data to give improved prediction of cell and spreadsheet error rates during review.

3. PROBLEM

Spreadsheet audits can require huge amounts of time and resources to conduct as mentioned by [Croll, 2003], “The time taken to review these models can range from twenty five hours to many hundreds, generating significant fee income for firms undertaking this work.”. If the level of error in the spreadsheet could be predicted, then a decision on whether the spreadsheets should be tested could be made. This would focus resources on the more error prone spreadsheets.

Information on the likely cell and spreadsheet error rate could be of great importance to management, allowing them to make a decision on whether a spreadsheet requires exhaustive testing or, indeed, whether it should be rejected and redeveloped. The prediction of likely cell and spreadsheet error rate, by an auditor, say, can be helped by expert and other knowledge such as developer experience or available audit information. The auditor can use this to aid the decision about the spreadsheet.

4. RESEARCH QUESTIONS

This work is part of a larger study and the particular question that we are trying to solve is can a model be established to predict the CER of large spreadsheets based on both expert knowledge and available test data. The predicted CER can then guide the decisions based on cell error rates.
The research first looked at literature on existing spreadsheet error research and general statistical methods for reliability including Bayesian methods. The relationship between the CER and the SER was investigated using statistical methods. This work was examined through the following questions:

**RQ1**: What does existing research say about the level of spreadsheet error and methods to discover errors in spreadsheets?

**RQ2**: What statistical methods can be used to predict cell and spreadsheet error rates?

The research completed for **RQ1** has looked at existing research into the level of errors in spreadsheets. This is described in Section 2. The research completed for **RQ2** first examined the relationship between the CER and the SER. A key issue in this regard is the nature of the dependence, if any, between cell errors. The research looks at defects, which are errors in formulas that produce an incorrect result. A dependence structure indicates that the probability of one cell containing an error influences the probability of any other cell containing an error whereas an independence structure assumes that no relationship between the error cells exists. In this work we make the critical assumption that unique cell formula errors are independent and we discuss this later the conclusion.

The research has also shown there are many different sources of information on spreadsheet error. The investigation aims to combine both prior knowledge and test data to estimate the CER of large spreadsheets. The possibility of creating a structure is explored through **RQ3**.

**RQ3**: Can a model be developed that combines prior knowledge and available test data to estimate the CER for large spreadsheets?

We propose the use of Bayesian methods that combine prior information and test data to answer **RQ3**. There are many external factors that can influence the CER of a spreadsheet. The developer and the organisation can impact the CER at the early stage of spreadsheet development. The complexity of the spreadsheet can naturally influence the error rate. The more complex the spreadsheet then the more likely it is to contain an error. The model will be evaluated using large spreadsheets once completed. The model will be tested to predict the CER with the results compared to the actual CER of the spreadsheets.

**RQ4**: How effective is the model at predicting the cell error rate in spreadsheets?

The fourth research question will be addressed in future work. The model will be evaluated using operational spreadsheets. The validity of the assumptions underlying the model will also be evaluated.

5. THE BAYESIAN MODEL

The model aims to estimate the CER of large spreadsheets by combining expert information and information on the correctness of individual cells gathered during partial review. The expert information is known as “prior information”. The spreadsheet may be partially examined. This information is called “test data”. Both sets of information can be combined using Bayesian methods to produce “posterior information”, from which error rate estimates can be derived.

5.1 Prior Information
The prior information can be based on a number of factors that influence the CER of a spreadsheet; the company, the developer, the spreadsheet complexity, etc. The characteristics of the developer and the company regulations play an early role in the life of a spreadsheet. The company environment could contribute to the CER as deadlines need to be met which can replace quality with speed. The developer may also not have the relevant skills or experience to complete the task which can produce more errors than an experienced developer. The spreadsheet complexity represents the level of complex calculations, formulas, and functions in similar/previous spreadsheets. These factors could indicate the level of error in the spreadsheet.

Further qualitative information based on initial structural and content investigation, often an initial step in a review methodology, can also influence the choice of prior distribution.

The prior information is represented as a beta distribution, the form of which is given below. This is the conjugate prior of the binomial distribution which represents the test data in section 5.2. The beta distribution for the cell error rate, $\theta$, has two parameters, $\alpha$ and $\beta$. These are set by the individual or individuals with best knowledge of this or related spreadsheets and represent the belief as to its likelihood for each of the possible cell error rate values. This expert may well be the reviewer who has already performed a cursory examination of the spreadsheet. In cases where very little is known about the spreadsheet a vague prior can be chosen which will reflect this lack of certainty regarding the likely error rate.

$$f(\theta) = \frac{\theta^{\alpha-1}(1-\theta)^{\beta-1}}{\int_0^1 \theta^{\alpha-1}(1-\theta)^{\beta-1}} \text{ where } 0 \leq \theta \leq 1$$

The Beta Distribution

The mean and variance of the distribution are as follows:

$$\text{Mean} = \frac{\alpha}{\alpha + \beta} \quad \text{and} \quad \text{Variance} = \frac{\alpha \beta}{(\alpha + \beta)^2 (\alpha + \beta + 1)}$$

The prior beta distribution can be used to produce an estimate of the likely CER before review begins based only on expert information which can be used to aid the decision as to whether detailed cell by cell testing is required. If the prior CER is not acceptable, then a decision to review the spreadsheet can be made with the possibility of further decision steps.

5.2 Test Data

The test data is information obtained by reviewing the cells in the spreadsheet individually. A binomial distribution can be used to model the occurrence of failures, errors, during this review. The binomial distribution returns the probability distribution of the number of $x$ successes (number of errors) in $n$ independent trials (number of tested cells) with probability $\theta$ of a success. The independence assumption addressed in section 6.1 is important,
\[ b(x;n,\theta) = \left( \frac{n}{x} \right) \theta^x (1-\theta)^{n-x} \quad \text{for} \quad x = 0, 1, 2, K, n \]

### 5.3 Posterior Information

The posterior information combines the prior information (expressed through a beta distribution) and the test data (modelled by a binomial distribution) to produce a statistical distribution which represents the likely values for the error rate for the cells that have not yet been tested. This is known as the posterior distribution. Using a beta prior distribution and a binomial model the posterior distribution also takes the form of a beta distribution, \( \text{beta}(+x, +(n-x)) \), the distribution for which is presented below.

\[
f(\theta) = \frac{\theta^{\alpha+x-1}(1-\theta)^{\beta+(n-x)-1}}{\int_0^1 \theta^{\alpha+x-1}(1-\theta)^{\beta+(n-x)-1}} \quad \text{where} \quad 0 \leq \theta \leq 1
\]

### 5.4 Beta Distribution of Prior and Posterior Information

The following section illustrates prior and posterior distributions. Suppose a spreadsheet contains 900 formula cells and expert information indicates a cell error rate of 0.2 with a Std. Dev. of 0.0146. This yields parameter values of 2 and 8 for the prior beta distribution. This information could also be considered as equating to having knowledge of 10 cells in the spreadsheet and finding 2 of these with errors.

Assume further that the first 20 formula cells are tested and the results found 2 error cells. This test data is combined with the prior, beta(2, 8), to produce a posterior beta (4, 26). The mean of the prior distribution is 0.2 with the mean of posterior distribution 0.13.

![Prior and Posterior Beta Distribution](image1)

![Prior and Posterior Information estimating Number of Error Cells](image2)

**Figure 1(a)** Prior and Posterior information in terms of Cell Error Rate and (b) Prior and Posterior probability estimates for number of error cells.

In Figure 1(a), the beta distribution is shown using prior information. The distribution suggests the likely CER is approx. 0.13 or 13%. The posterior information suggests the
likely CER is approx 0.12 or 12%. This is almost double the prior’s probability. The shape of the distribution has now changed from being wide around the mean for the prior to taller and thinner for the posterior. This shows our belief that the posterior information is more certain then the prior information. A narrower prior distribution represents a stronger belief in the cell error rate.

Figure 1(b) shows the likely number of error cells of the remaining 880 untested ones based on both prior and posterior distributions. The prior distribution shows that the number of errors is likely to range from 0 to 550 approximately with 110 the most likely number of error cells. The posterior distribution indicates a smaller range of 0 to 320 likely errors approximately 96 errors cells.

6. POSTERIOR INFORMATION

This section illustrates the methodology through its application to three operational financial spreadsheets. The spreadsheets relate to research project finance in a third level institute. The spreadsheets were audited by the authors using the [Powell, 2008] methodology. The analysis was conducted by two people independently. The results were compared and agreement reached over the errors found.

The sequential analysis of the formulas begins in the first worksheet in the first unique formula block. This block is checked for errors and the result is recorded. The process continues onto the next formula, column by column and then row by row in a sequential process. An example of how the data used from the error analysis is shown in Table 1.

<table>
<thead>
<tr>
<th>Cells Tested</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
<th>6</th>
<th>7</th>
</tr>
</thead>
<tbody>
<tr>
<td>No. Errors</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>2</td>
<td>3</td>
</tr>
</tbody>
</table>

Table 1: Example of Results

As formula cells are tested, the numbers of errors are recorded. Assuming a prior with parameter values of (5, 95) is selected beforehand, the mean of the posterior error rate distribution is updated as each unique formula block is checked. The mean is calculated using the formula in section 5.1 and it decreases as no errors are found and increases for discovered errors. Each time a unique formula block is checked a new posterior mean is calculated.

6.1 Unique Formulas

Given that the standard measure of error rate is based on cells, we could base our model on individual cells rather than blocks of copied cells. However, the model makes the important assumption that the relationship between error cells is independent. However copied formulas contradict this assumption as the original cell would influence the error status of the copied cells. Therefore the analysis looks at unique formulas in spreadsheets.
A spreadsheet section which contains 12 non-empty cells; 3 formula cells and 9 data input cells can be seen in Figure 2. Two of the formulas are copies, cell B5 and cell D5, of the formula in A5. The first unique formula is A5 and can involve more than one cell, if the formula cells are directly above, below, in front or behind with no blank or different value cell between. In Figure 2, there are 2 unique formulas cells, A5 and D5. This is different to root formulas which are measured throughout the spreadsheet, [Panko, 2007].

6.2 Applying to Operational Spreadsheets; Defects.

This section presents the analysis of three operational spreadsheets focusing on a distribution for the level of error remaining in the spreadsheet. The results are in terms of unique formulas and with focus on defect errors. The following figures show the best estimate for the defect error rate (through the mean of the posterior distribution) as testing proceeds including 5% and 95% limits which indicate how wide the distribution for the error rate extends. All the graphs are produced based on prior parameters of (5, 95).

Reliability graphs are also included which gives the probability that the error rate for the remainder of the spreadsheet is below a selected acceptable error rate chosen in this case to be 5%. A firm may decide that they will stop the review if they are happy that this acceptable defect rate has not been breached.

Spreadsheet I

In Figure 3(A), the posterior mean shows periods where unique formulas contain defects. These periods are shown by the sharp jumps in the mean. The defect free unique formulas are shown by the gradual descents in the graph. The graph also contains two lines either side of the mean. The top curve represents the 5% chance that the CER is greater than this line. The bottom curve gives the error rate for which there is a 5% chance of the true value being below. There is a 90% chance that the true error rate lies between the 5% and

![Spreadsheets I](image-url)
95% curves. The areas between these outside lines; which include the mean, imply 90% confidence that the CER lies in this interval.

Some organisations may allow a certain level of error to remain in the spreadsheet. If the actual error rate falls below this acceptable CER, then the organisation may not deem the spreadsheet to need further examination. The reliability calculates the likelihood of the posterior CER being equal to or below the acceptable CER. In Figure 3(B), the reliability that the defect error rate stays less than or equal to the acceptable CER of 5% sharply decreases with some areas of improvement. The reliability spikes sharply and decreases very closely to 0% for 50 – 100 unique formulas. The reliability begins to increase towards the end and finishes with 20% reliability. This suggests the spreadsheet requires further testing or redevelopment as the reliability is very poor and management cannot be sufficiently sure that the error rate is below an acceptable level. The same reasoning can be applied to the following Figures.

Spreadsheet II

Figure 4: (A) The posterior mean for defects with confidence interval. (B) Defect reliability

Spreadsheet III

Figure 5: (A) The posterior mean for defects with confidence interval. (B) Defect reliability

In Figure 5(A), the posterior defect mean starts at the prior information and reduces until defects are found. The posterior information improves the prior information as the mean reduces and becomes stable to produce conclusions. If the prior information was not included we would see a zero mean until defects were found and would create large jumps. This would prove difficult to produce any conclusions based on the information as it is not stable. In Figure 5(B), the reliability increases and stays above 95% from 100
unique formulas until the end of the spreadsheet. This would indicate after, say 100 – 130 unique formulas have been tested, that testing can stop based on the defined criterion. We believe this to be true by applying this method to improve spreadsheets.

7. CONCLUSION

This paper outlines research that examines the use of Bayesian methods to combine test result data with expert knowledge on the likely error rate to yield posterior estimates of reliability that can support managers in their decision as to how much of a spreadsheet needs to be tested. This method may be particularly suited to large spreadsheets for which it can be very costly and time consuming to review the entire spreadsheet.

There are a number of issues with the method. Firstly, the method assumes perfect discovery with respect to spreadsheet error during the review, although research has shown this to be extremely difficult. Secondly, the method does not factor in omission errors into the prediction. Thirdly, we assume that errors are independent. While we record errors based on unique formulas to avoid the clear issue of dependence between copied cells, there are other ways in which cell errors may be independent. Cells that contain the same or related functions may be related in terms of likelihood of error. Likewise, cells in close proximity in a spreadsheet may not be independent in terms of likelihood of error. This can raise questions as to how well review results for one part of a spreadsheet can be used to predict the reliability of another part of the same or a different spreadsheet. A more natural way of achieving better representative coverage of the spreadsheet and to overcome this issue might be to randomly choose the formulas to check. This is the subject of ongoing research.

Finally, error rates are only one aspect of spreadsheet quality measurement. As Powell states “obtaining reliable estimates of cell error rate is only a first step toward understanding the fundamental question of the impact that spreadsheets have on the quality of decision making in organisations”, [Powell, 2007a].

8. Acknowledgements

This work is supported by the Technological Sector Research measure funded under the NDP and ERDF.

9. REFERENCES


Cockram, T. (2001), "Gaining Confidence in Software Inspection Using a Bayesian Belief Model." Software Quality Control, Vol. 9(1); 31 - 42.


THERE WILL BE A BETTER WAY . . .

Angus Dunn  
(angus@thedowerhouse.fsnet.co.uk)

Few major commercial or economic decisions are made today which are not underpinned by analysis using spreadsheets. It is virtually impossible to avoid making mistakes during their drafting and some of these errors remain, unseen and uncorrected, until something turns the spotlight on them. By then it may be too late. The challenge is to find a way of creating spreadsheets which will preserve the benefit of their power and flexibility while making their creation more transparent and safer. Full documentation and documented version and quality control, section by section, of the eventual spreadsheet would be a bonus. And if the whole process could be made quicker, too, that would be a further bonus.

1 WHERE WE ARE NOW

Practitioners know from bitter experience how difficult it is to produce spreadsheets that are wholly reliable. Professionals have pointed out the high proportion of spreadsheets that contain material errors [KPMG 1997]. Academic studies indicate that virtually all spreadsheets contain errors [Panko 1998, revised 2008]. [Pettifor 2003] was confident in saying that “the world is not falling apart through spreadsheet errors” - in 2003 it wasn’t falling apart. In 2009 things are rather different and in due course history will allocate the proportion of blame attributable to bad spreadsheeting. The financial sector, at the root of the collapse, is a major user of spreadsheets. “Within certain large sectors [of the financial sector] 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” and “Spreadsheets have been shown to be fallible, yet they underpin the operation of the financial system” [Croll 2005].

Spreadsheets tend also to suffer from lack of detailed specification and poor maintainability [Pryor 2004], lack of comprehensive (or, indeed, of any) documentation, lack of modular design and inadequate testing, to name but a few.

Even if history holds bad spreadsheeting entirely blameless for the present mayhem, the fact that spreadsheeting is so widely used while at the same time so error prone is a strong indication that if a way could be found to improve reliability, global GNP would step noticeably upwards.

In a quarter century as a project banker working with and alongside spreadsheets, the author became increasingly frustrated at the failure of new tools to emerge to improve their reliability. As a result, he set out to develop one himself.
2 THE AIMS

A spreadsheet should be easy to read [Raffensperger 2001]. A brief glance should be enough to indicate to the reader where the inputs are, where the calculations are, where the internal checks are and which pieces contain the outputs designed to be printed out as hard copy.

The inputs and the calculations should be modular [Grossman and Özlük 2004] , each section representing a logically discrete portion of the total computation. The subject matter of each section should be clear, and it should be obvious to the reader as he scrolls through which section he is looking at at any time.

Related formulae should be in physical proximity; and the spreadsheet should read from left to right and from top to bottom [Raffensperger 2001]. In the calculations and inputs, related topics should where possible be recorded adjacent to each other; inputs should be recorded in the same order as are the calculations which draw on them.

If there is the slightest doubt about how the spreadsheet treats each section of the analysis, or if the spreadsheet uses one of a number of possible assumptions about how a particular item or set of items is calculated, this should be documented; the spreadsheet will generally be more readable if there is also an appropriate level of descriptive documentation as an aid to finding the way around.

If a data book will ultimately be required, any text entries which are required to describe individual sections should be drafted at the same time as the sections themselves.

There should be a record of who drafted each section and when.

Each section, when first drafted and after every change, should be independently checked; documentation should show whether this has been done, by whom and when.

There are no prizes for needless reinvention of the wheel; logic once written and proven to work should be available for use within the same firm in future spreadsheets which analyse the same circumstances, until that logic is itself superseded.

The spreadsheet should have a detailed specification.

The spreadsheet should be maintainable.

The hard coded parameters of the spreadsheet - start date, width, periodicity - should be input at the very end of the process, not at the very start, thus allowing these to be varied at will, without requiring changes to the template.

Finally, it goes without saying that the rules generally accepted by practitioners for safer spreadsheeting should be followed. Inputs, calculations and reports should be separated, multiple scenarios of inputs should be capable of being run through the same computation and there must be confidence that all inputs are brought – once and once only - into the calculations [Grossman and Özlük 2004], use of constants should be discouraged in formulae, all formulae in the calculations areas of the spreadsheet must be left to right consistent, there must be extensive use of self checking, circularity should be avoided where possible, and controlled and tested for convergence of result where avoidance is impossible.
3 THE ANSWER

The author’s answer, provisionally called RingtailXL, is a Windows application which provides a framework for generating spreadsheets. Once its template is complete, the spreadsheet, its databook and its complete specification can all be generated.

The advantages claimed for RingtailXL, in comparison with handcrafting spreadsheets directly, include:

- Substantial saving of time
- A big increase in flexibility
- A big increase in transparency
- Radical improvements in quality control
- The ability to maintain complete documentation of all the constituents of final spreadsheets
- Enforcement of good practice in the separation of data, computation and output and enforcement of left to right consistency in formulae

Together, these should result in far fewer errors.

If these claims prove robust, major spreadsheet users would experience radical improvements in productivity, consistency and quality.

4 HOW

RingtailXL enables users to assemble templates to generate spreadsheet models. The templates consist of three different elements, called Components, Skeletons and Models.

A Component is a representation of a small group of adjacent rows to be placed on one of the calculation sheets of the final spreadsheet, designed to model a particular aspect. Indeed, when the final spreadsheet is generated, these lines are shown “Grouped”.
A Component may contain one or more “Embedded Components”, thereby allowing the same logic to be used several times but with different inputs or a complex task to be broken into a number of sub-tasks.

Once sufficient Components have been prepared to undertake all the tasks the final spreadsheet needs, these are strung together into a single “Skeleton” of the calculation area(s) of the final spreadsheet; it is the collection of these separate Components which provide the resulting spreadsheet with its “modularity”.

Additionally, in the Skeleton, the structure of each data input, the width of all the formulae (ie, full width, single column or special) and the checks which the spreadsheet will undertake, are all defined. If many models of similar structure are made for each particular type of project, a unique skeleton would probably be constructed for each such type.

The Model imports a single Skeleton, and requires the user to complete the remaining structural details - details of the data structures, all special formula widths, and display formats for the output of each formula - and enables the user to enter multiple data scenarios and to define output sheets, in the form of reports and charts.

It also enables the user to define additional calculation sheets, and specify on which such sheet each formula in the spreadsheet will appear. Thus, though the default structure of the resulting Workbook is for all the calculations to be placed on a single worksheet called “Workings”, additional calculation sheets can be generated by users who prefer a different style. The system enforces separation into one or more input sheets, one or more calculation sheets, a sheet of checks and sheets which report output, in the form both of reports and charts.
From a higher level, an element at a lower level can be drilled into, changed, saved as a new version and the new version substituted for the old version in the element at the level above. Thus from a Model, its Skeleton can be called up and changed, and from a Skeleton any Component can be called up and changed, and from a Component, any Embedded Component can be called up and changed. After returning to the level above, information and parameters stored at that level which are still relevant are retained; information which is no longer relevant is discarded.

Each element has a feature allowing it to be diagnosed for completeness and for certain errors.

From the Model, a full Excel Workbook, a detailed specification and a databook can all be generated.

The whole spreadsheet development cycle should be undertaken within RingtailXL; during its lifetime many versions of the spreadsheet will probably be generated, most of which will be discarded. Provided the templates have been kept, any version of the spreadsheet can be re-created at any time, together with its specification and databook. The intention is that manual changes are never made to the spreadsheets themselves and indeed these all contain a feature which shows whether each is “clean” or has been changed by hand since it was generated.
5 DOCUMENTATION

Each element (the Components, the Skeletons and the Models) stores four pieces of documentation.

“Notes” and “Data Book Entry” are input by the user. The Notes can be used for any purpose. The Data Book Entry shows what a Data Book, generated at the end, will contain in respect of the element concerned.

The other two pieces of documentation, “Status” and “Audit Trail” are generated by the system.

The Status has three possible values – OK (this element has not been materially changed since it was last checked), Warning (this element has been materially changed on one occasion since it was last checked) and Failure (this element has been changed from one which previously showed a Warning or a Failure). The Status of any element also records the Status of all elements contained within it. Icons on the main RingtailXL window indicate the Status of the currently active element.

The Audit Trail shows the version from which the element was derived and the version as
which it has been saved and lists all the changes (whether considered material by the system or not) between the two versions.

6 CONCLUSIONS

The work on RingtailXL leads to the conclusion that it will be possible to use spreadsheet generators to draft spreadsheets that are modular, easy to read, fully documented, fully specified, demonstrably checked, and which need no maintenance. The work also holds out the hope that the process of developing such spreadsheets will be less stressful and faster than is the process of hand-crafting directly into Excel.

Whether the ideas contained within RingtailXL will take off as the basis of such a spreadsheet generator remains to be seen, but they certainly could. RingtailXL is actively looking for interested partners to help take it into the mainstream.

When such Workbook generators are generally used, they will make the world a safer place. And not only will they make it safer – by increasing the reliability of the tool almost universally used to underpin decision making, they will make it richer too. Which will perhaps offset to an extent the blame history is likely to lay at spreadsheeting’s door for the way the world is in 2009.
REFERENCES


Revised 2008 - http://panko.shidler.hawaii.edu/SSR/Mypapers/whatknow.htm


Croll, G. J. (2005), “The Importance and Criticality of Spreadsheets in the City of London”, European Spreadsheet Risks Interest Group 6th Annual Symposium


An approach for the automated risk assessment of structural differences between spreadsheets (DiffXL)

John Hunt
Excel for Managers
John.Hunt@ExcelForManagers.com

ABSTRACT

This paper outlines an approach to quantify the risks associated with changes made to spreadsheets. The methodology focuses on structural differences between spreadsheets and suggests a technique by which a risk analysis can be achieved in an automated environment. The paper offers an example that demonstrates how contiguous ranges of data can be mapped into a generic list of formulae, data and metadata. The example then shows that comparison of these generic lists can establish the structural differences between spreadsheets and be used to quantify the level of risk that each change has introduced. Lastly the benefits, drawbacks and limitations of the technique are discussed in a commercial context.

1 INTRODUCTION

In business, a primary reason for choosing to use a spreadsheet over a fully managed computer application is the spreadsheet’s flexibility: - Its ability to be updated or adapted quickly. Actual figures, for example monthly sales data, are often expected to change, but the calculation or formulae in a spreadsheet may also need regular review or up-dating to meet new requirements, legislation or even the ability to correctly adapt to a new financial year.

In many organisations, changes to the workings and calculations of a spreadsheet are left relatively unmanaged. A requirement to create new documentation or test the changes is often left to the conscience of the spreadsheet owner. Consequently managers, operational risk managers or auditors are frequently left unaware that a modification has even been made.

It follows that in many instances a change in a spreadsheet cell value cannot be accurately attributed to an expected numerical change (say sales data) or a change caused by a difference in functionality, or both. Therefore this ability to make unregulated changes to a spreadsheet may by its own nature give rise to a business and audit risk that is unknown, un-quantified and quite possibly simply ignored.

Left unmanaged, a spreadsheet that was once well designed, documented, tested and implemented, may well become prone to significant unknown errors over time.
2 BACKGROUND

2.1 Definition: Structural Change

In the context of this paper, a structural change is defined as the addition, removal, or modification of any formula or data in a worksheet cell. A structural change does NOT include a change to the format of a cell or a change in the displayed/calculated value of a previously defined formula cell.

2.2 Why test for structural changes?

Many spreadsheets used for reporting or valuation purposes are re-run on a cyclical basis. Good business practice dictates that any change to a spreadsheet's process or calculation method is documented and, where appropriate, suitable re-testing and/or approval carried out. However it is not unusual for undocumented and untested changes to creep into spreadsheets, often accidentally. A typical example of an accidental change is the replacement of a formula with an absolute value to mitigate a known error in source data. If the spreadsheet is subsequently saved, a structural change has been made, and future values have a high risk of being incorrect.

2.3 Minimising risk & semi-automated testing

Post-design, there are several simple ways in which businesses can help minimise the risks from spreadsheet changes. Some of these are physical methods, such as locking formula cells and protecting worksheet structure. Others are process driven such as a peer review of changes, the requirement to run pre-defined test scripts after each change, or using a spreadsheet comparison program to highlight value differences from identical input data.

2.4 Comparing spreadsheets: Formula view

One useful test/auditing method is to examine the formula view of a spreadsheet. The visibility of text or values entered directly (such as in B2:B4 below) is maintained, whilst the formulae themselves are shown (such as in D2) instead of their resultant value.

Reviewing formulae and values in Excel™ 2007
A key reason for viewing spreadsheets using this method is that direct structural spreadsheet comparisons can be made.

Given that there are no other differences between the spreadsheets (such as the use of different macros or named range changes etc.), it is safe to conclude that for any given set of inputs two spreadsheets with identical cell-for-cell formula views will give the same results: - Many spreadsheet comparison programs use a coded version of ‘Formula View’ as their primary basis of comparison.

2.5 Auditing, testing & reviewing with a formula map

Formula mapping programs are auditing tools that give a visual representation of the formula and data-structure of a worksheet, which can be useful after the initial spreadsheet design or after any known changes have been made. They make it relatively simple to spot unexpected formulae variations and some inconsistencies such as the accidental overwriting of a formula cell with a value.

Formula maps generally split a spreadsheet range into elements. In the example below, cells of the same colour represent cells with the same R1C1 formula, and static, directly entered values are shown in grey, making it relatively easy to spot the accidental overwriting of a formula with a number, as highlighted.
3 A SUGGESTED APPROACH FOR THE AUTOMATED RISK ASSESSMENT OF SPREADSHEET CHANGES

3.1 Definition: DiffXL

DiffXL is the creation and subsequent comparison of contiguous range data and metadata, with a view to providing a risk based analysis of structural spreadsheet differences.

3.2 Creating Generic DiffXL Data Lists

This section proposes a method that can be used to ascertain if there are certain differences between spreadsheets or spreadsheet versions, and attempts to categorize the relative level of risk that the change has created.

Please note that the terminology, samples and references within this paper relate to a Microsoft Excel™ implementation of the methodology. Although the terminology may differ, the principles should be identical in other spreadsheet applications.

The proposed method first involves the storage of generic data from contiguous-range elements within each worksheet. Each element is deemed to consist of a rectangular range of cells with the same formulae (in R1C1 reference style) or a rectangular range of cells with no formula. The elements’ details are stored for future comparative analysis. The actual data stored about these elements may take many forms, each with subtly differing analysis requirements and conclusions available. One such structure is described below by way of an example.

Consider the following mini-spreadsheet, shown below in normal, formula map and R1C1 formula view. The steps to create the rectangular element list are as follows …
The first ‘rectangular’ element is established. In this example it consists of six cells of non-formula data, highlighted. Note that although the first row is evidently a title, and rows 2 to 6 are other entries, they are both considered as a single rectangular element. The start row and column are recorded, along with rectangle’s size, recorded as the number of rows and number of columns. There are no formulae for this rectangle, so the letter ‘V’ (for Value) is entered and the formula column is left blank.

(R1C1 reference view of the spreadsheet) (Generic list of metadata)

The first rectangle element is now excluded, and the next rectangle is established. In this case it is the remainder of the title row. Its position and size is noted as before.

Note that in this particular example the rectangles are being established using an algorithm that broadly creates segments top to bottom and then left to right. The order in which the segments of any contiguous range are established is unimportant, but the method must be consistent for future comparisons to be accurate.

The third rectangle element is a series of 5 formulae, whose R1C1 formulae are identical. The position and size is recorded. An ‘F’ (for formula) is noted, along with the R1C1 formula.

The 5 remaining elements are calculated and recorded. Note the inconsistent value in row 4, column 3 which prevented a single large area (R2C2 to R6C3) being created.
3.3 Recording static information

In order to ascertain if any change between spreadsheets has been made, there is also a requirement to assess any change in static information. To give a guaranteed 100% accuracy then both spreadsheets must be opened and a cell by cell comparison made, or every static cell’s location and value must be stored. Assuming the preferred option is to store the values in the generic list, this can be achieved as follows…

By definition, each rectangular segment (line in the list) must consist of either formulae or static data. For each segment that contains static data, concatenate the cells’ string values and separate every cell with a delimiting character (for example ‘|’) irrespective of the cell’s status. In the previous example, the first segment could therefore be made to include the data string…

Base|CORONA|CORONA|CORONA|CORONA|CORONA

and this string could be stored in the Formula column, i.e.

This exacting method may be appropriate or essential in some circumstances. However it can become cumbersome for large segments, and result in generic list file sizes approaching or even exceeding that of the original spreadsheet.

One alternative is to concatenate the cell values as above, but hold a checksum based on the entire string. Although it would not be possible to ascertain which cell(s) had changed directly from the checksum, it would still be possible to ascertain with a high degree of probability that a change had been made within any data segment, and if necessary the two files opened to retrieve the difference by cell.

A small but important limitation for storing static values in this way is that any cells containing a value consisting solely of the separating character or characters (e.g. || in the above example) could fool a change-test algorithm. In the commercial implementation of this method, a non-printing character is used as the separating character to help minimise this risk.

A final example of a segment list for a worksheet with two contiguous ranges is shown below.
3.5 Drawing risk based conclusions from the comparison of two DiffXL lists

In order to draw appropriate conclusions, a comparison of two lists is required. The expectation is that this process would be accomplished automatically through a series of difference algorithms. Both qualitative and quantitative conclusions can be made.

At this stage, it is also possible to include a subjective risk assessment of any changes. The actual and genuine level of risk is dependent on numerous factors, including the type of data, type of change, and number of simultaneous changes.

The category of risk also needs to be taken into account. For example risk may be considered in different terms if the user is looking at the risk of an incorrect result, the risk of a catastrophic malfunction of the spreadsheet or the risk of failing an audit, and so on. A variety of change-test algorithms, or settings within the algorithms would therefore be deemed to be appropriate within different industries or departments.

Below are some of the types of change to spreadsheet structure that can be ascertained using DiffXL, together with a suggested (user-configurable) associated level of risk.

<table>
<thead>
<tr>
<th>Comparison</th>
<th>Risk</th>
</tr>
</thead>
<tbody>
<tr>
<td>Entire range moved</td>
<td>Low</td>
</tr>
<tr>
<td>New static text, placed away from other ranges</td>
<td>Low</td>
</tr>
<tr>
<td>One or more ranges sorted</td>
<td>Low</td>
</tr>
<tr>
<td>Text or headers changed</td>
<td>Medium</td>
</tr>
<tr>
<td>Contiguous range split into two or more segments</td>
<td>Medium</td>
</tr>
<tr>
<td>New items added within a range</td>
<td>Medium</td>
</tr>
<tr>
<td>Formulae change throughout an entire range</td>
<td>Medium</td>
</tr>
<tr>
<td>Items removed from within a range</td>
<td>Medium</td>
</tr>
<tr>
<td>Items removed or changed, breaking up a range</td>
<td>Medium</td>
</tr>
<tr>
<td>Numeric cell changed to a number which is formatted as text</td>
<td>High</td>
</tr>
<tr>
<td>Overwrite a formula with a number or text</td>
<td>High</td>
</tr>
<tr>
<td>New data and formulae introduced</td>
<td>High</td>
</tr>
<tr>
<td>New or deleted worksheets</td>
<td>High</td>
</tr>
</tbody>
</table>
4 BENEFITS, DRAWBACKS & LIMITATIONS OF USING THE DiffXL METHODOLOGY

4.1 Benefits of using the DiffXL methodology

The ability to demonstrate that no structural differences have been made to a spreadsheet over time, or to be able to identify where changes have been made, may be of benefit to some spreadsheet owners, managers and auditors.

DiffXL’s ability to assign a relatively low risk indicator to simple changes such as the movement of a block of cells from the top to the bottom of a spreadsheet, or the sorting of a data range, means testing and approval effort can be best allocated where changes have been identified as more likely to result in material differences to calculated values.

DiffXL can be used in commercially sensitive or secure environment: Once a DiffXL list has been created for a spreadsheet, that list is generally the only requirement to perform difference tests or analysis. If the checksum method for storing static values is used, there are therefore few security issues with releasing the lists for external analysis or reporting.

A Diff XL based control framework can be run as a background process, either directly when a spreadsheet change is made or at any pre-defined frequency. A system could easily alert owners or managers only if a change of a certain magnitude of risk has been detected. This offers the potential for a discreet and un-obtrusive corporate testing regime.

4.2 Drawbacks & limitations of using the DiffXL methodology

Changing spreadsheet data or formulae is only one way to change the operation of a spreadsheet. In a commercial application, other aspects of the spreadsheet also need to be considered and analysed, for example changes to macros, named ranges or data connections.

Unless direct cell-to-cell comparison between spreadsheets for static values is used, there is a small risk that a value change will not be detected by this methodology. In most implementations the risk is negligible, but nevertheless possible and dependent on the method in which the static data is stored (directly or through a checksum), the quality of any checksum, and the use of the separating character.

Although based on objective algorithms, overall risk assessments may still be subjective. As an example, should three low risk changes constitute a low, medium or high risk change?
5 DiffXL IN COMMERCIAL APPLICATIONS

Listed below are screenshots from two products where the DiffXL methodology has been applied in conjunction with more established analysis techniques.

a) An auditor’s tool that directly compares spreadsheets from one year to the next to ascertain the level of risk associated with any changes. The tool can be run directly from a memory stick so no software installation is required and gives a quick indication on whether (or where) further detailed analysis is required.

b) A SharePoint™ Control Framework that monitors spreadsheets and alerts the owner if cumulative changes have exceeded a pre-defined level of risk. Here, the spreadsheet’s complexity and change-risk at each ‘publish’ is charted for the owner or auditor. A non-SharePoint™ version of this product is also available.
Further details of the DiffXL Audit Assistant and the IDAS SharePoint Control Framework can be obtained from the author.

ACKNOWLEDGEMENTS

The author wishes to express sincere thanks to Mike Lines of Rule Financial Ltd for his assistance with the commercial implementation of DiffXL.

6 SUMMARY

This paper has discussed an alternative spreadsheet comparison methodology.

DiffXL does not compare spreadsheets directly and is not a substitute or replacement for any of the established auditing and comparison methods and tools.

For some owners, risk managers or auditors, it may be speedier or relatively more important for them to understand the risk profile of the difference rather than the difference per se. For these people a comparison based upon DiffXL can offer insightful information.

A DiffXL toolkit or control framework may be of particular interest to auditors who wish to quickly highlight areas of most concern and to companies that desire discreet ongoing spreadsheet control management.
Documenting Problem-Solving Knowledge: Proposed Annotation Design Guidelines and their Application to Spreadsheet Tools

Matthew Dinmore
Department of Information Systems, University of Maryland, Baltimore County
1000 Hilltop Circle, Baltimore, MD 21250
mdinmo1@umbc.edu

ABSTRACT

End-user programmers create software to solve problems, yet the problem-solving knowledge generated in the process often remains tacit within the software artifact. One approach to exposing this knowledge is to enable the end-user to annotate the artifact as they create and use it. A 3-level model of annotation is presented and guidelines are proposed for the design of end-user programming environments supporting the explicit and literate annotation levels. These guidelines are then applied to the spreadsheet end-user programming paradigm.

1. INTRODUCTION

End-user programmers’ principal goal is to solve problems, not write programs. However, the mechanics of solving problems through programming – even when the user is not intentionally writing code, as in traditional programming – often take precedence over documenting the solution. The resulting software artifact contains within it the tacit problem-solving knowledge gained in the process of not only creating it, but of putting it into practical use. Because this knowledge is rarely made fully explicit, reusability as both a software and knowledge artifact is reduced.

A significant challenge is finding a means to document problem-solving knowledge in an unobtrusive, seamless manner. If the cost of capturing the knowledge is greater than the perceived benefit, a user is unlikely to do it. This is largely a matter of design; regular users of a tool will know the challenge of deciphering an artifact they themselves created, so there is certainly incentive to document for themselves. Those engaged in collaborative problem-solving endeavors have an incentive to share, and a design that enables, at a minimum, the kind of simple, in-process note-taking common to this work [Markus, 2001] is a worthy objective toward satisfying this requirement.

An apt place to start is with the spreadsheet. It is by far the most popular end-user programming tool [Jones et al., 2003]; its widespread use in the financial industry and other domains has led to the recognition that errors can be costly [Panko, 1998]; these are certainly software errors, and as such, there are efforts to bring software engineering-like practices to the creation of spreadsheets and other end-user development media [Burnett et al., 2004].

A common solution that potentially satisfies both requirements – to capture knowledge and reduce errors – may be found in designs supporting structured annotations. Knowledge is made explicit through annotation [Marshall, 1998] and there is evidence that structured combinations of code and documentation reduce errors and improve
maintainability [Oman and Cook, 1990]. The objective of this work is to combine and extend these ideas in the end-user development space, with an initial focus on the spreadsheet paradigm.

This paper begins by scoping the work around problem-solving knowledge transfer and typical methods of achieving this in software development. A model of end-user development annotation is introduced as a framework for a collection of guidelines for integrating annotation as a means of knowledge documentation into the end-user programming paradigm. These guidelines are then applied to the spreadsheet end-user development modality, and a plan for evaluating the resulting designs is described. Finally, related work is compared to the proposed approach.

2. PROBLEM-SOLVING KNOWLEDGE

This work focuses on improving the transfer of problem-solving knowledge. This is formally defined as the ability to apply previously learned knowledge to solving a new problem [Mayer and Wittrock, 1996]. In this process, prior learning helps new learning; this effect is increased when the prior learning – that is, the hard-won problem-solving knowledge learned by another – is codified in such a way to make it useful to someone taking on a new problem. A common approach to this is analogical problem solving, in which the solution to a previous, but analogous, problem is tailored to the new problem. The challenge then becomes in finding analogous problems and solutions to reuse.

The degree of codification of the knowledge is closely related to the concepts of tacit, implicit and explicit knowledge [Polanyi, 1966; Wilson, 2002]. Briefly, tacit knowledge is that which is unexpressed and largely inexpressible; implicit knowledge is partially expressed but may require additional knowledge – some “common sense,” at least as far as the expresser of the knowledge is concerned – to understand it; and explicit knowledge is fully stated and generally reusable as is. Many challenges face efforts to effectively represent and share knowledge, and these are often beyond the purely technical, requiring engineering in the larger socio-technical domain [Fischer and Otswald, 2001].

To scope this challenge, a useful point of departure is to identify the elements of problem-solving knowledge that need to be communicated. As adapted to computer-based problem solving by Mayer [2002], there are four kinds of knowledge of interest to knowledge transfer: declarative, conceptual, procedural and metacognitive. For a software artifact, generally only the procedural and perhaps elements of the declarative knowledge associated with a program are represented within it; capturing the other elements requires additional documentation.

3. INCREASING LITERATENESS OF KNOWLEDGE EMBEDDED IN SOFTWARE

In traditional programming, a number of mechanisms have been employed to capture programmer knowledge, principally various forms of documentation internal and external to the source code. Certainly the most common method is the use of comments within the code, and consequently, comments are the most relied upon source of knowledge for programmers attempting to understand previously written code [Souza et al., 2005; Das et al., 2007]. However, programmers are inconsistent in how they write comments [Marin, 2005], and efforts to structure comments have largely failed. An alternative is to provide separate (from the source code) documentation; this approach suffers from a gradual divergence from the source, as the documentation is often not maintained [Forward and Lethbridge, 2002], and as such, programmers rely on these artifacts much less than source code comments [Souza et al., 2005].
In both cases – adding structure to comments and maintaining external documentation – the challenge is a human one: how are programmers to be encouraged to maintain documentation at a level that promotes effective reuse? An approach that has emerged over the last decade is the use of automatic documentation generators, an example of which is Javadoc. This system takes comments embedded within the code and creates a hyperlinked document describing the classes and methods of a Java software artifact [Kramer, 1999]. However, as Raskin [Raskin, 2005] argues, automatically generated comments likely encourage developer laziness by giving the appearance that the documentation has been written, and he appeals to programmers to take to heart Knuth’s essay on literate programming [Knuth, 1992] as the “...gospel... for all serious programmers” regarding documentation.

The literate programming approach, so named by Knuth because it engenders the perspective of software code as a form of readable literature, suggests a different approach to development, one in which the author of a program communicates the meaning of the program to another person rather than simply to a computer to execute. This is achieved by embedding segments of code within a narrative structure that describes them. Two key concepts intrinsic in this are chunking and verisimilitude. Chunking of logical code segments allows the author to describe in a natural ordering (rather than, for example, a compilation or execution ordering) the flow of the program, perhaps in the context of how it solves a problem. Chunking along logic boundaries in traditional programming has been shown to improve programmer comprehension [Norcio and Kerst, 1983]. Verisimilitude is the unification in a single artifact of the code and the documentation, which, as previously discussed, results in improved synchronization and maintenance of the documentation.

Research into the efficacy of literate programming has demonstrated positive, if occasionally anecdotal, results. Despite this promise, literate programming certainly has not become commonplace; requirements such as learning an additional markup language and a lack of programmer-friendly tools integrated into the software development cycle have been cited as impediments [Cordes and Brown, 1991]. Recent developments in Integrated Development Environments have begun to address these criticisms, and new thinking about the application of literate programming has led to modernized approaches [Pieterse et al., 2004]. Holmes [2003] offers the concept of perspicuous programming that extends the literate paradigm by explicitly incorporating documentation for a program’s users within the software, that is, the software itself becomes the reference manual. When applied to the domain of end-user programming, in which the developer and user are increasingly synonymous, the resulting artifact becomes a unified one containing the software, its design rationale and the problem-solving knowledge behind the solution it encodes, and end-user literate programming is realized [Dinmore and Norcio, 2007].

4. MODEL OF ANNOTATION IN END-USER PROGRAMMING

As a means toward achieving end-user literate programming, this work adopts the concept of annotations, which are information provided in addition to that which is critical to the core function of an artifact; annotations here are not restricted to text, but may be “rich,” containing any kind of media. Marshall [1998] identifies several dimensions for annotation including tacit to explicit and informal to formal. Tacit annotations represent incomplete thoughts, and are normally written as “memory joggers” intended to be personal and therefore useful primarily to the author. At the other end of the scale, explicit annotations contain all of the information necessary to understand the annotated thought.
With the goal of enabling increased literacy in end-user programs, a collection of design guidelines for supporting rich annotation have been formulated for the designers of end-user programming environments. As a framework for explicating these guidelines, a model of annotation in end-user programming is offered (figure 1):

![3-level design model of annotation in end-user programming.](image)

This model describes three levels of annotation: implicit, explicit and literate. The implicit level applies to annotations that are largely unwritten (as text) or only represent partial thoughts, with the rest of the author’s thought remaining tacit [Marshall, 1998]. As an example in the spreadsheet paradigm, consider the programs represented by figure 2:

![Implicit (left) and explicit (right) annotation in spreadsheets.](image)

The first spreadsheet is implicitly annotated: it contains its formulae – accessible only by clicking on each cell – as well as some column headings and a sheet title. There are no instructions for using the sheet, nor are there extended descriptions of the computations.

The second sheet in figure 2 is explicitly annotated. The explicit level demonstrates intent on the part of the developer to create meaningful annotations. Many authoring environment offer mechanisms for adding text or rich, multi-media annotations that exist outside the primary medium, though the question is whether these ad hoc tools are effective. In particular, it is often left to the author to decide how to implement these to create truly communicative annotations. The results will vary. Here, the spreadsheet developer has used labels, structure, narrative text, and even color.
However, there are still some problems with this. In order to achieve this level of annotation, the developer had to enter the narrative text in the paragraphs on separate lines, manually wrapping the text (see the paragraphs starting at row 138). This is a constraint of the annotations being bound to the underlying structure of the spreadsheet. A possible solution is to use the commenting feature of the spreadsheet, but this again presents its own set of limitation: the annotation remains hidden until activated, it is bound to a cell rather than existing as an independent object, and when displayed, it covers a portion of the spreadsheet.

Finally, the literate level of annotation fully adopts the literate programming paradigm, and in doing so, changes the nature of the development environment to one that is composed of a linear flow of chunks of different types. Recall that the ordering of the chunks is based on the best way to describe the program rather than any notion of an execution or compilation order. This design is easily extended to include other chunk types, for example images or other rich media: multimedia has been shown to improve comprehensibility of procedural knowledge texts [Stone and Glock, 1981].

To establish the extent to which implicit and explicit annotation are present in current spreadsheet usage, a survey of a large collection of spreadsheets – the EUSES corpus [Fisher and Rothermel, 2005] – is underway. Manual sampling (n=104) preliminary to a complete survey indicates that, of those spreadsheets that are computational (41%), 42% are implicitly annotated and 58% are explicitly annotated. Of the explicitly annotated spreadsheets, about half (28% of the total) exhibited literate-like levels of annotation and chunked complexity, though they clearly were not constructed in the literate paradigm. The relatively more sophisticated use of annotation indicates a desire among users to annotate their spreadsheets, supporting the direction of this work toward developing better annotation-enabling features.

5. PROPOSED ANNOTATION GUIDELINES

Given this model, design guidelines for the informed-explicit and literate annotation levels can now be enumerated. These guidelines are briefly described here, and then subsequently applied in more detail to the spreadsheet paradigm.

These proposed guidelines break into two groups. The first, encompassing guidelines 1-7, are overall guidelines for the design of annotation-supportive environments. Guidelines 8-11 suggest features that can enhance and leverage the presence of rich annotation to improve the user experience and thereby support increased adoption.

5.1 Document problem-solving knowledge in the most accessible manner

This guideline is the most fundamental, pragmatically calling for the use of notations that are readily accessible to the vast majority of users; in general, this means the use of textual narrative and images rather than some more compact, though less accessible notation, such as a purely mathematical or machine-centric notation. Rich text and multimedia support improved knowledge transfer [Agresti, 2000].

5.2 Address all levels of end-user development

Mücher [1997] defines three levels of end-user tailoring: customization, integration and extension. Customization involves the ability to change parameters in a program (for example, values in the cells of a spreadsheet), while integration involves adding structures within the notation (adding a new spreadsheet within a workbook or new collection of formulas). Extension is about adding new features to the environment, for
example through a plugin interface or macro programming language. Supporting this spectrum of users and needs is necessary for a tool to be successful.

5.3 Seamlessly maintain documentation and software code

This is an expression of the concept of verisimilitude [Wyk, 1990] and recognizes that unifying the software and narrative-expressed knowledge in a single, logical artifact is more effective than attempting to maintain multiple artifacts. Note that this does not require this to be done in a single logical artifact, but rather only at the presentation level. However, it is important to consider management of the artifact beyond the runtime environment; a single logical artifact is likely easier for users to manage.

5.4 Manage annotations and code as first class objects

Handling annotations in the same manner as the software elements places them on equal footing and presents to the user a common interaction paradigm [Marshall, 1998]. It also allows for increased interoperability among objects and a closer mapping to the users’ mental model [Repenning and Ioannidou, 2006]. Borrowing from literate programming, this is accomplished by viewing all objects as chunks.

5.5 Arrange annotations in a structured manner

This guideline implements the finding that book-like presentations improve software comprehension and maintenance [Shum and Cook, 1994]. The linear design presents a logical flow, though in this case, the logical ordering may be in the explanation rather than the computation. This guideline is in response to explicit but uninformed designs in which the annotations have no particular unifying structure, a situation which might detract from comprehension.

5.6 Support global visibility and ease of navigation

Literate programs enable the automatic creation of a table of contents, cross-reference and index. Recalling the benefits of a hierarchical structure for problem-solving, and implementing the user interaction guideline of overview plus details on demand [Shneiderman, 1997], this guideline calls for the interface to include these automatically generated navigational mechanisms.

5.7 Support multiple audiences with differing needs

End-user developed software may serve a variety of users with different objectives ranging from neophytes who need to learn the domain, to advanced users who simply want to run a computation. These users will require different interfaces, and in the case of annotated software, different levels of annotation. This can be accomplished by facilitating themes [Kacofegitis and Churcher, 2002] and enabling chunks to be assigned to different, named themes. A user can select a theme, which then presents only the chunks in that theme. Additionally, ordering in themes can be different, allowing for different logical paths through the artifact.

5.8 Provide scaffolds for effective problem solving and task execution

This is the first of several higher-level guidelines, implemented on the foundation provided by the preceding guidelines. Here, it is recognized that certain structures – templates or compositions of components – can address particular use cases more effectively. For example, it may be useful to adopt a software patterns structure for
certain kinds of problems. This would involve a templated arrangement of chunk types that would be tailorable, of course, but provide a scaffolding with which to start. Worked problems [Renkl et al., 2000] and task-specific designs are included within this guideline.

5.9 Encourage documentation by automatically identifying and creating placeholders for items to annotate

Even when well intentioned, a user is often more focused on problem-solving than documenting. By identifying key elements to annotate and creating documentation stubs for them, the user is both reminded of the need for documentation and intelligently assisted in selecting the pieces of the software artifact to document. This extends the behavior of documentation generators, such as JavaDoc [Kramer, 1999], into end-user programming.

5.10 Prevent software errors

Bringing software engineering methods to end users is challenging, particularly when one objective of end-user programming is to make the programming aspect essentially invisible. This guideline calls for the value-added use of the structures and rich content contained in the artifact to help prevent errors. Common features in modern programming environments, such as syntax coloring and autocompletion should be implemented. Additionally, annotations can allow for the expression of assertions as part of the documentation, and these can be actively checked at runtime [Burnett et al., 2003].

5.11 Provide integrated, intelligent access to reusable assets

Recommender systems can be employed to actively suggest reusable elements, thereby leveraging the rich semantics inherent in the annotations – both in the problem-solving domain and in the software domain – that may be relevant to the user’s work. This also supports the development of communities; end-user programmers benefit from the artifacts produced by others and the social interaction in this process [Nardi, 1993].

6. ANNOTATION-SUPPORTIVE SPREADSHEETS

In this section, the above guidelines are applied to the spreadsheet modality of end-user programming. Spreadsheets are an attractive choice for an initial design-in-use evaluation: First, they are by far the most common end-user programming tool [Jones et al., 2003]. Their interface semantics are relatively uniform across varying spreadsheet applications, so the guidelines will be widely applicable. Finally, they have been the subject of a significant amount of research that can be leveraged toward this objective.

The guidelines are applied here at both the informed-explicit and literate levels of the end-user programming annotation model. The key difference is that, at the explicit level, the familiar spreadsheet application is used as the foundation and features supporting annotation consistent with the guidelines are added on top of that foundation. At the literate level, a completely different application foundation (from the users’ perspective) is introduced, though as argued by Sajaniemi [2002], users’ familiarity with spreadsheets and text editing makes the leap much less intrusive.
The informed-explicit design consists of a traditional, multi-tabbed spreadsheet stack positioned next to a multi-tabbed document pane; this pane can be closed, making the system resemble the traditional spreadsheet and potentially easing transition to the environment. The annotations pane allows for different, themed sequences of rich annotation chunks to be presented. A hyperlinked table of contents can be revealed for quickly navigating to a topic. Formulas can be defined within the text and then used in spreadsheet, providing a low barrier to rapid reuse for the user and encouraging adoption.

Extending this to the literate design, the spreadsheet pane is removed and now the spreadsheet components become another chunk type within the themed views. While a simple change to make in the design, it fundamentally changes the user’s view from one in which the annotations are an easily removable — and therefore ignorable — addition to the spreadsheet, to one in which they are fully integrated.

These designs drive certain requirements for the calculation environment. For example, because there are multiple chunks, it is necessary to provide a global referencing mechanism. Properly implemented, this allows for a number of unusual features, including the ability to define formulas within a chunk-type of their own, so that the formula is entirely visible, annotatable and reusable by reference; and to reference spreadsheet ranges from within the text. The spreadsheet execution paradigm is an excellent match with the literate design, because it allows for the chunks to be presented in any order. Since calculations only rely on the reference graph, the presentation can be completely decoupled from the computation.
7. EVALUATION

Studies are in progress to evaluate designs based on the proposed guidelines as compared to traditional spreadsheets; the initial research is designed to explore the efficacy of the proposed guidelines in improving knowledge transfer and reuse. To accomplish this, the Location-Comprehension-Modification model of reuse [Fischer et al., 1991] provides a framework for an experimental study in which subjects are measured in each of the aspects of this model across four conditions: implicit annotation (traditional spreadsheet with no annotations), uninformed-explicit (traditional spreadsheet with use of built-in commenting tools and in-cell annotations), informed-explicit (a new, guideline-informed design based on the spreadsheet paradigm), and the new, literate design. Subjects will perform three experiments: the first will measure the relative findability [Azzopardi, 2008] of artifacts in a corpus (“location”, where the hypothesis is that rich annotation increased intrinsic findability using common search engines). The second will measure comprehension using classical techniques, such as simulation [Dunsmore and Roper, 2000]. Finally, subjects will perform a maintenance task to measure their ability to modify an artifact. It is expected that the individual and aggregate results from each sub-
experiment will be measurably different among conditions, with the design interventions demonstrating better results. Should these results support the claim of increased reuse, future studies will focus on optimizing the interaction mechanisms for creating rich annotations, thereby further refining the guidelines and offering specific, experience-based implementation tradeoffs to software designers.

Throughout the initial work, and certainly in the annotation creation interaction-specific work, the Cognitive Dimensions of Notations framework [Green, 1989] will be used throughout to guide the designs and their evaluation from a notational usability perspective. This framework is appropriate in that it has been demonstrated to effectively address end-user programming and, more generally, the abstract idea of notations.

8. RELATED WORK

There are a number of previous research efforts focused on improving the documentation of spreadsheets. Several authors approach the challenge in the context of existing spreadsheet tools, primarily Microsoft Excel, offering recommendations for structures and best practices in support of documentation. Payette [2008] describes a methodology for applying the built-in “comment” functionality to create structured annotations. Powell and Baker [2007] present a chapter on spreadsheet engineering best practices, among them “designing for communication” and “documenting important data and formulas.” Similarly, Raffensperger [2008] references the structure of traditional programs in introducing a similarly-inspired structure for spreadsheets, summarized in four general guidelines, including “format for description, not decoration” and “expose rather than hide information.”

Sajaniemi [2002] describes a prototype implementation of a fusion of the the traditional spreadsheet and word processor. This design presents a unified user interface that naturally combines many of the features to which users are already accustomed. Among the functionalities that would need to be reconsidered for this combined paradigm are the display of results, the indexing notation, and referencing mechanism.

Paine [2008] offers a proposal to extend the Excelsior [Paine et al., 2006] spreadsheet design language with literate programming-style documentation constructs. Excelsior is inherently modularized, and as such, fits the chunked style of literate programming well. This approach is more in keeping with the original processing model for literate programming, as well: programs written in Excelsior exist as separate text files that are then used to generate a spreadsheet and, in the case of Literate Excelsior, documentation. In the present work, there is no structurally separate processing phase.

9. CONCLUSION

This work seeks to improve the capture and transfer of problem-solving knowledge in end-user programming. Efforts to date have focused on the explication of a collection of design guidelines for seamlessly incorporating explicit annotation into various end-user programming modalities, with an initial focus on the widely-used spreadsheet. These design guidelines draw heavily from the literatures of human-computer interaction, software engineering, knowledge management and design of communications, as well as pragmatic studies of the spreadsheet paradigm, and are strongly influenced by Knuth’s literate programming. Design prototypes are in the process of being iteratively improved and an experimental evaluation of the designs with respect to their impact on artifact reusability and user satisfaction is planned.
10. REFERENCES


Burnett, M., Cook, C., Pendse, O., Rothermel, G., Summet, J., & Wallace, C. (2003), End-user software engineering with assertions in the spreadsheet paradigm, Portland, Oregon.


Holmes, N. (2003), The case for perspicuous programming, Computer, 36, pages 102-104.


Marin, D. P. (2005), What Motivates Programmers to Comment?, EECS Department, University of California, Berkeley.


Raskin, J. (2005), Comments are more important than code, Queue, 3(2), pages 62-64.


An Exploratory Analysis of the Impact of Named Ranges on the Debugging Performance of Novice Users

Ruth McKeever, Kevin McDaid, Brian Bishop
Software Technology Research Centre, Dundalk Institute of Technology, Dundalk, Ireland
catherineruth.mckeever@dkit.ie, kevin.mcdaid@dkit.ie, brian.bishop@dkit.ie

ABSTRACT

This paper describes an exploratory empirical study of the effect of named ranges on spreadsheet debugging performance. Named ranges are advocated in both academia and industry, yet no experimental evidence has been cited to back up these recommendations. This paper describes an exploratory experiment involving 21 participants that assesses the performance of novices debugging a spreadsheet containing named ranges. The results are compared with the performance of a different set of novices debugging the same spreadsheet without named ranges. The findings suggest that novice users debug on average significantly fewer errors if the spreadsheet contains named ranges. The purpose of the investigative study is to derive a detailed and coherent set of research questions regarding the impact of range names on the debugging performance and behaviour of spreadsheet users. These will be answered through future controlled experiments.

1 INTRODUCTION

Lack of regulation in the financial sector is now more topical than ever. The introduction of legislation such as Sarbanes-Oxley, which enforces financial controls on spreadsheets in businesses in the US, has focused minds on the issue of spreadsheet error. These errors are partially attributed to the distinct lack of professional developers involved in creating spreadsheets, as the majority of spreadsheets are created by the actual users. A further issue is the lack of structured methodologies. One study found that only 6% of development time is spent testing spreadsheets [Baker et al, 2006]. Another [Powell et al, 2007a] found errors in 94% of spreadsheets, and 1-2% of cells. The single largest error found by [Powell et al, 2007b] had an impact of greater than $100 million.

Many attempts have been made to develop tools and best practices that would reduce the high error rates found in spreadsheets, such as WYSIWYT (What You See Is What You Test) [Rothermel et al, 2000], and U-Check [Abraham & Erwig, 2007]. Despite these efforts the instances of material errors continue to occur, both accidentally and intentionally. As an example, in 2005 a sorting error caused aspiring police officers to be incorrectly informed that they had passed an exam [EuSpRIG, 2009].

This research seeks to establish if the use of named ranges could make a spreadsheet easier to understand, and therefore easier to debug. The experiment detailed in this study examines the performance of novice users asked to debug a spreadsheet, seeded with errors, that makes extensive use of named ranges (all the formulas in the spreadsheet use names). The authors also question how frequently named ranges are used in real-world spreadsheets, and examine a repository of sample spreadsheets in order to answer this.
The motivation behind this study is research into how refactoring methods, as used in software engineering, can be applied in practice to support the development of better quality spreadsheets. The goal of the work is to generate tighter research hypotheses and questions to guide future research as to the merits or otherwise of named ranges in spreadsheet technology.

Refactoring is a feature of Agile Software Engineering. It is a technique for cleaning code by making small changes that improve the internal structure of the code, without changing the external behaviour of the program [Fowler M. 1993]. These small changes make errors easier to find, as the code becomes more understandable. Two important Refactoring methods are *Rename Method* and *Replace Magic Number with Symbolic Constant*. *Rename Method* is centred on the principle that a method name should be changed to reflect its purpose. Developers are encouraged to think about what a comment would say about the method, and rename the method accordingly. Likewise, a formula in Excel can be renamed to reflect its purpose. The *Replace Magic Number with Symbolic Constant* method aims to eliminate the unnecessary hard coding of numbers into software, as this practice frequently leads to bugs. Instead the number should be assigned to a variable, which can then be changed in one place instead of throughout the code. Constants in a spreadsheet can be defined in exactly the same way so that they can be changed in one place rather than in every cell that uses them.

Spreadsheet engineering is more closely aligned to agile rather than document driven development processes as utilised in the creation of computer software. It values working software over documentation – often omitting documentation entirely, and can easily respond to change. [Baker et al, 2006] Cite [Cragg and King, 1993] that over 85% of spreadsheets had been modified after their initial implementation and that models are updated an average of 7 times. This is the motivation for using agile practices in an attempt to improve the quality of spreadsheet development.

1.2 Overview

This paper has the following layout. Section 2 details the background research beginning with a description of named ranges, an explanation of what the authors regard as quality in spreadsheets, and the research questions that guided this study. Section 3 describes the methodology used to explore the research questions. Section 4 describes the results of the investigative experiment. Section 5 contains a discussion of the results, covering possible causes, the limitations of the study, and the new research questions developed. Section 6 concludes this paper.

2 BACKGROUND RESEARCH

2.2 Named Ranges

A range is an individual cell, or group of cells. By naming a range it can then be referred to in formulas throughout the spreadsheet in the same way that a variable is named in software code. By giving a range a meaningful name, as one would give to a variable or method in code, it is believed that formulas will become clearer to the user, therefore more understandable and testable. Without meaningful range names the user must remember the meaning of a cell named, for example “H79” and then check for its occurrence in formulas throughout the workbook. E.g. the formula “netIncome =
grossIncome – tax”, at first glance is far more understandable than “E80 = A40 – D69”. To name a range the developer simply highlights the range and enters a suitable name in the name box (the box above and to the left of the worksheet that normally contains the reference of the cell that is currently in focus).

The inclusion of the Name Manager in the Excel 2007 ribbon greatly improves the ease with which the user can add, modify and delete names. It also provides the facility for the user to sort and filter the names, and provides a quick insight to what each range refers. While typing a name in the name box is a simple way of creating a range name, it is not possible to change either the name, or the values to which it refers, in this location. The Name Manager tab gives the developer quick and easy access to these type of modifications. It also allows the developer to associate further information with the name, in the form of a 255-character comment.

To go to a named range in a workbook the user can either click on the name box, or press F5 on the keyboard, and select which range they wish to go to. This brings the focus directly to the named range. The user can also insert a list of all the named ranges into a worksheet by pressing F3 and choosing Paste List. A developer can name a single cell, a group of cells, a constant, or a formula. Used properly names can be a powerful tool with many properties and uses, including the following:

**Inserting a reference:** By naming a cell or group of cells you can insert that range elsewhere in the spreadsheet simply by referring to the name. Because names are set as absolute by default, the reference will not change. If the original cells are subsequently moved the name will still refer to the same values, hence the inserted values will remain correct. If the values to which the reference refers change, then the inserted values will also be updated.

**Different levels:** Names are most commonly used at workbook level, but they can also be declared at sheet level. This can be useful so that one name can be used refer to the same range of cells on several sheets.

**Absolute and relative referencing:** Names are absolute by default, but when creating a named formula the developer has the option of using absolute or relative referencing. If the developer wishes to use relative cell referencing it is important that they are in the cell that the formula will initially be used in, as it is from this cell that the reference will refer.

**Constants:** Constants can be named without needing a cell reference. In the Name Manager dialog box the developer simply defines a name, and in the Refers To field enters a value rather than a range or cell reference.

**Validation:** Named ranges can be used for data validation by naming a range of allowed values and then using this name as the source for the validation list.

**Dynamic named ranges:** Dynamic named ranges allow the developer to name a range where the size of the range is not known from the outset, or if the size may change. This is implemented with use of the OFFSET and COUNT functions to calculate how many values are contained in a range and then setting the name accordingly.

Other technologies have been developed to support the user in the management of range names. One example is OPERIS Analysis Kit (OAK) [OPERIS, 2009] which includes features to modify names to correct misspellings, apply and delete multiple range names
and, most interesting in the context of the results we will present later, the ability to replace range names with cell references.

2.3 Quality in Spreadsheets

Quality of spreadsheets in this study refers to reliability, understandability, testability and extendibility. The reliability of a spreadsheet is essentially the accuracy of the data that it produces, and is compromised by the errors found in approximately 94% of spreadsheets. Understandability refers to how easily a user or auditor can make sense of the spreadsheet, and is fundamental to the implementation of Sarbanes-Oxley. Testing is crucial if the reliability of the spreadsheet is to be proven, yet is next to impossible if the spreadsheet is not understandable. Extendibility relies on the previous characteristics, yet is vitally important considering how frequently spreadsheets are reused and remodelled.

2.4 Research Questions

The review of literature related to the use of Named Ranges in spreadsheets was guided by the following questions:

• Research Question 1 (RQ1): Do industrial, academic, standards and training organisations advocate the use of named ranges, and if so, how should they be used and why?
• Research Question 2 (RQ2): To what extent and in what way are range names used in practice?

RQ1: Do industrial, academic, standards and training organisations advocate the use of named ranges, and if so, how should they be used and why?

The SSRB (Spreadsheets Standards Review Board) [Hutchens, 2005], Microsoft [Microsoft Corporation, 2006], and Read & Batson (1999) advise the use of named ranges. In their paper for IBM [Read & Batson, 1999] the authors state that “allocating meaningful range names to areas or cells within a spreadsheet can speed up the development process, make the model easier to understand and reduce the risk of errors made by referring to the wrong cell.” They also suggest naming constants rather than hard-coding them into cells, as this makes them easier to change, and recommend referring nearby cells by cell reference and far-away formulas by range name. In their 2006 white paper [Microsoft Corporation, 2006], Microsoft recommend “named ranges to reduce errors and increase formula readability.” The SSRB, in their Best Practice Spreadsheet Modelling Standards [Hutchens, 2005], describe detailed naming principles, including four naming conventions related to range naming, “Every range name in the workbook should describe the content or use of the range being named”. They provide a list of prefixes that should be used with different types of names.

Range naming is recommended by many websites devoted to spreadsheet advice [Pearson, 2007], [Mr Excel, 2007], [Ozgrid, 2008] and [Johnson, 2008], and by Microsoft in their online Excel documentation [Microsoft, 2008]. As far back as 1985 the Journal of Accountancy ran an article [Bromley, 1985] in which the author states that defining names for cell ranges “reduces the probability of cell reference errors caused by moving a column or row to another location.” [Bewig, 2005] advocates the proper construction of range names for eliminating the problem of referring to the wrong cell while constructing formulas, and states that “well-chosen names are the first and best form of documentation.”
In an article on Spreadsheet Accuracy Theory [Kruck & Sheetz, 2001], the authors propose that naming ranges can help the developer comprehend cell meaning “As cells become more highly interconnected, the developer spends inordinate time trying to remember the meaning of the cell, and she/he is distracted from using it effectively. Naming of ranges [Miller 1989] and structure [Ronen Palley Lucas 1989] helps with this problem.” The author sums this research by stating “simple policies requiring separation of data areas and user interface areas, requiring cell naming, limits on formula length, or documentation of tests conducted would result in more accurate spreadsheets”.

One survey was conducted of spreadsheet users in Australia [Hall M. 1996], which found that 60% of the surveyed participants used named ranges. The author then stated that 75% of the participants said that they should have used named ranges, however there was no elaboration as to why they felt that they should be used.

Range naming is not without its detractors. The authors of [Panko & Ordway, 2005] warn against the dangers of a range name referring to an incorrect range, and therefore appearing correct when it is not. They state “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.”

The author of [Blood, 2002] advises against using named ranges, stating that they are unnecessary if the developer has designed the layout well. Three main disadvantages of range names are set out in this paper. Firstly, very descriptive range names lengthen simple formulas, which should already have their function made clear by a row label. Secondly, range names hide the location of the cells to which they refer, and thirdly they create “ghost” links when sheets are copied to a different workbook.

Despite this conflicting advice from spreadsheet experts, there is no academic research at present that examines whether the use of names in spreadsheets increases quality.

RQ2: To what extent and in what way are named ranges used in practice?
To establish how named ranges are currently used, a quantitative evaluation of existing spreadsheets was developed. This analysis was carried out on the EUSES Corpus [Fisher M. 2005], a repository of 5606 real-world sample spreadsheets, including 4498 unique spreadsheets, available to spreadsheet researchers. The authors found that 51% of these spreadsheets contained names, however not all these names referred to ranges. When studied in more depth it was found that of those spreadsheets that do contain names, 46% use these names exclusively for defining print areas, and only 2% of them use names in formulas. The remainder of the spreadsheets contained names that referred to tables, names that linked other workbooks or databases, appeared to either have no function or had their reference deleted.

2.5 Investigative Research

After reviewing the literature, it was decided to examine the impact of range names on debugging performance. An investigative experiment was designed to explore how novice users perform debugging a spreadsheet seeded with errors that was developed using range names wherever possible. This experiment is described in the following section.

3 METHODOLOGY

The experiment used to evaluate the hypothesis was based on research previously conducted by the authors of [Bishop & McDaid, 2007]. It involves asking a group of
novice spreadsheet users to examine a spreadsheet and to correct any errors that they found. The experiment described in [Bishop & McDaid, 2007] is used as the comparative group for this study. This approach has several limitations, which are discussed in section 5.

### 3.2 Original Experiment

This original experiment was described in [Bishop & McDaid, 2007] and compared the debugging capabilities of expert versus novice spreadsheet users. It was based on another study conducted by [Howe & Simpkin, 2006] and consisted of a spreadsheet containing three sheets: Payroll, Office Expenses and Projections. Payroll calculates the payroll expenses for a typical week, Office Expenses sums office expenses for the first quarter and estimates the same for the remainder of the year, and Projections estimates the total expenses, both office and payroll, for the next 5 years. This spreadsheet was seeded with 42 errors, and contained no range names.

While the group debugged the spreadsheet, T-CAT, a “time-stamped cell activity tracking tool”, [Bishop B., McDaid K. 2008] ran in the background recording data about each cell click. This tool was developed as a macro in VBA in order to record the interaction between the participant and the spreadsheet with minimal intrusion. This data recorded included each cell entered, at what time each cell was entered, and what changes were made. The resulting data is printed to a hidden worksheet when the spreadsheet is closed and can then be analysed to examine each participants’ interaction with the spreadsheet.

The errors seeded in this trial can be divided into four categories: Clerical/Non material, Rule Violation, Data Entry and Formula. There were four Clerical errors such as spelling mistakes, and four Rule Violation errors where the data in the spreadsheet violated the company policy as detailed in the instructions sheet. There were 8 Data Entry errors, for example where a number was entered incorrectly. The Formula errors were further divided into four sub-categories: logic, cell, range and remote. Logic errors consisted of illogical formulas, and instances where a formula was expected but a number used instead. Cell reference errors consisted of errors where an incorrect individual cell was referenced while range errors consisted of errors where an incorrect group of cells were referenced. Remote reference errors occurred where an incorrect reference to a different sheet was used.

### 3.3 Experiment on Named Ranges

For our study the original structure and format of the spreadsheet was retained, as were the 42 seeded errors. The only change was the implementation of named ranges. Individual cells were named rather than arrays, because of the difficulties faced when attempting to change a value in an array.

This experiment group consisted of 21 students in their second year of a computing degree in Dundalk Institute of Technology, who had a year earlier taken a class in spreadsheet basics such as formulas and charts. This experiment was carried out in accordance with the ethics policy in Dundalk Institute of Technology. The participants were first given a tutorial on naming ranges, composed of a presentation and a practical exercise that asked them to create, edit, use and delete range names. The authors monitored their performance in this introductory task and were satisfied that they were able to use named ranges to the extent required to perform the experiment. As shall be shown, their performance in the task clearly supports this assumption. When this was
completed satisfactorily, they were asked to open up the experimental spreadsheet and begin the trial.
The group consisted of a mixture of Irish and international students. There was some concern that the meaning of the range names might not be clear to the students for whom English is not their first language. The initial analysis separated the students but as there was no significant difference in performance between the different nationalities, the following results are based on the group’s performance as a whole.

Each student was given an instructions sheet detailing the rules and assumptions that the data in the spreadsheet was expected to follow. They were asked to correct the errors directly on the spreadsheet. The students were not given a time limit for the trial, although it was expected that they would take no longer than an hour.

4 RESULTS

The results showed that the students corrected on average 47% of the errors seeded in the spreadsheet. This is approximately 11% less than the comparative group. Unsurprisingly, the only statistically significant difference in performance relates to the debugging of formula errors, where the experiment group corrected only 44% of errors, while the control group corrected 63%.

<table>
<thead>
<tr>
<th>Error Type</th>
<th>No of Seeded Errors</th>
<th>% Corrected by Experiment Group</th>
<th>% Corrected by Control Group</th>
<th>Experiment Compared to Control</th>
</tr>
</thead>
<tbody>
<tr>
<td>Clerical/Non-Material</td>
<td>4</td>
<td>11%</td>
<td>11%</td>
<td>0%</td>
</tr>
<tr>
<td>Rule Violation</td>
<td>4</td>
<td>63%</td>
<td>65%</td>
<td>-2%</td>
</tr>
<tr>
<td>Data Entry</td>
<td>8</td>
<td>64%</td>
<td>63%</td>
<td>1%</td>
</tr>
<tr>
<td>Formula</td>
<td>26</td>
<td>44%</td>
<td>63%</td>
<td>-19%</td>
</tr>
<tr>
<td><strong>Total</strong></td>
<td><strong>42</strong></td>
<td><strong>47%</strong></td>
<td><strong>58%</strong></td>
<td><strong>-11%</strong></td>
</tr>
</tbody>
</table>

The similar results in the Clerical, Rule Violation and Data Entry categories indicate that the control group and experiment group are comparable in terms of ability and knowledge. As the experiment group performed significantly worse in the formula errors, these were examined further to establish which type of formula errors posed the most problems.
<table>
<thead>
<tr>
<th>Error Sub-Type</th>
<th>No of Seeded Errors</th>
<th>% Corrected by Experiment Group</th>
<th>% Corrected by Control Group</th>
<th>Experiment Compared to Control</th>
</tr>
</thead>
<tbody>
<tr>
<td>Logic</td>
<td>9</td>
<td>54%</td>
<td>63%</td>
<td>-9%</td>
</tr>
<tr>
<td>Cell</td>
<td>7</td>
<td>39%</td>
<td>68%</td>
<td>-29%</td>
</tr>
<tr>
<td>Range</td>
<td>7</td>
<td>47%</td>
<td>71%</td>
<td>-24%</td>
</tr>
<tr>
<td>Remote</td>
<td>3</td>
<td>19%</td>
<td>28%</td>
<td>-9%</td>
</tr>
<tr>
<td><strong>Total</strong></td>
<td><strong>26</strong></td>
<td><strong>44%</strong></td>
<td><strong>63%</strong></td>
<td><strong>-19%</strong></td>
</tr>
</tbody>
</table>

Figure 2 - Formula Error Correction Results

While the experiment group performed worse in all the formula sub-categories, the greatest difference was with *Range* and *Cell* reference errors, and these were the only groups where the results were statistically significant at a 5% level based on a non-parametric rank sum test. In each case of these types of error the control group performed better than the experiment group.

These are the precise category of error that range names were expected to reduce, as these errors are all due to either an incorrect name used in a formula (e.g. the formula in *Payroll F11* is "=GriffinPayRate * HartfordRegularHours" instead of "=GriffinPayRate * GriffinRegularHours"), or a name referring to the wrong cell or range (e.g. the formula in *Office Expenses F18* is "=SUM(FixedYearEst)", but this name refers to an incorrect range of cells).

5 DISCUSSION

5.2 Causes

The results were a surprise to the authors given the widespread opinion that range names improve the understandability and reliability of spreadsheets. We next investigate three possible explanations.

High cognitive load

Because the participants were debugging a spreadsheet that they had not personally developed, it was thought that they would have trouble remembering what each name referred to, and might spend additional time switching back to check this, especially if the range used was located on a different sheet. They would have to complete two checks, one to see if the correct range name was used, and another to see if the name referred to the correct range. This would result in a higher cognitive load and thus may explain the poor performance. However, the students were taught how to click on a name in the name box (or by pressing the F5 key) and it would bring them directly to that range. Unfortunately it is unclear how many of them used that facility, as this was not recorded by T-CAT.

Too much confidence in names
Because the students were given a tutorial on naming ranges before the experiment commenced, it is possible that they saw the benefits of names and therefore expected them to be correct. This is difficult to prove, but it appeared that on a students’ initial reading of a formula, if the first range name appeared to be correct then they did not inspect any further. For example, an error contained in Payroll F9 occurred when the working hours of one employee were multiplied by the pay rate of a different employee. Only 14% of the experiment group corrected this error, in comparison with 65% of the control group. The erroneous entry was “=EnglebertPayRate*DanielsRegularHours”. The correct entry should have been “=EnglebertPayRate*EnglebertRegularHours”. 19% of the experiment group, and 65% of the control group corrected a similar error on the same sheet. It seems that when the participants saw that the first name appeared correct (to the extent that the name was what they expected to see) they presumed that the formula was correct.

The experiment group performed significantly (23%) better in one Formula Logic error, where a number had been used instead of a formula. This indicates that the error was immediately obvious because it contained no names, and was not consistent with other cells that performed the same task.

**Did not understand the error or did not know how to correct it**
One threat to the validity of this experiment was the possibility that the participant might not fully understand the concept of names. As they were second year computing students they understood the concept of variables. Before the participants began the trial each one watched a presentation and completed a task that taught them how to use range names. This task involved creating new named ranges, creating names to refer to constants, using these names in formulas and editing existing formulas so that they used names instead of cell references. Only when the authors were satisfied that the students could carry out these tasks were they asked to complete the experiment. This ensured that they understood the concept of names.

To investigate whether there was evidence that participants had insufficient knowledge to utilise range names, we examined in detail how each participant handled each range formula error, for which the debugging performance was comparatively low. In some cases it was thought that the subject might have been able to identify the error but unable to repair it. We considered that this would result in numerous unsuccessful attempts at debugging the cells. As Figure 3 shows, while up to 7 attempts were made to correct range errors, of those who attempted these errors the majority succeeded in correcting them. In two cases a participant failed after one attempt, and in one case a participant failed after two attempts. This reinforces the authors’ belief that the participants had sufficient knowledge about naming in order to correct the errors.
## Range Errors

<table>
<thead>
<tr>
<th>Error Location</th>
<th>Corrected on Attempt No.</th>
<th>Failed on Attempt No.</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>1</td>
<td>2</td>
</tr>
<tr>
<td>Payroll G16</td>
<td>8</td>
<td>1</td>
</tr>
<tr>
<td>Payroll H16</td>
<td>6</td>
<td>1</td>
</tr>
<tr>
<td>Payroll I10</td>
<td>5</td>
<td>5</td>
</tr>
<tr>
<td>Payroll I14</td>
<td>8</td>
<td>2</td>
</tr>
<tr>
<td>Office Expenses F10</td>
<td>6</td>
<td>1</td>
</tr>
<tr>
<td>Office Expenses F18</td>
<td>5</td>
<td>1</td>
</tr>
<tr>
<td>Projections G22</td>
<td>4</td>
<td>1</td>
</tr>
</tbody>
</table>

Figure 3 Range Reference Errors

Figure 3 shows the location of the range errors, how many participants corrected, or failed to correct, each error, and after how many attempts.

An error in *Office Expenses F18* referred to an incorrectly defined name on a different sheet. 43% of the experiment group, and 74% of the control group corrected this. This cannot be attributed to a lack of knowledge regarding how to fix the error, as of the 10 participants who attempted to fix the error, 9 succeeded. This was the only error where a range was incorrectly defined.

### 5.3 Limitations

An alternative approach might have been to randomly split the 21 participants into experimental and control groups. However, the small number of participants made this unworkable in terms of deriving statistically significant results. Instead, we argue below that the performance of the entire group can be compared with the group from the previous study.

Figure 4 compares the results of three trials: the experiment described in this paper (Group A), the original experiment used as the control group (Group B), and the control group from another experiment (Group C). The novice users in the original trial consisted of thirty-four second year accounting and finance students. The third trial had been carried out on fourth year Software Development students as part of another study [Bishop B., McDaid K. 2008].
<table>
<thead>
<tr>
<th>Error Type</th>
<th>No of Seeded Errors</th>
<th>% Corrected by Group A</th>
<th>% Corrected by Group B</th>
<th>% Corrected by Group C</th>
</tr>
</thead>
<tbody>
<tr>
<td>Clerical/Non-Material</td>
<td>4</td>
<td>11%</td>
<td>11%</td>
<td>13%</td>
</tr>
<tr>
<td>Rule Violation</td>
<td>4</td>
<td>63%</td>
<td>65%</td>
<td>66%</td>
</tr>
<tr>
<td>Data Entry</td>
<td>8</td>
<td>64%</td>
<td>63%</td>
<td>66%</td>
</tr>
<tr>
<td>Formula</td>
<td>26</td>
<td>44%</td>
<td>63%</td>
<td>63%</td>
</tr>
</tbody>
</table>

**Figure 4 Comparison of Results by Group**

Because of the consistencies between the groups of students that had already partaken in this experiment, and as this study is exploratory, the authors deemed the novice users from the original experiment to be suitable for use as a control group.

Another major limitation of this study is that its sole focus is on novice users. The effect of named ranges on expert users cannot be presumed, as it has been proven that professionals are significantly better than novices at correcting particular types of errors [Bishop & McDaid, 2007].

The students chosen for this study were computing students and therefore already understand the concept of variables. Spreadsheet users from a different background are unlikely to understand this principle so easily.

### 5.4 Future Work

The results suggest clearly that range names can have a detrimental affect on spreadsheet debugging. It is important to use the results to provide more focused and coherent research questions that can be answered through further PhD study by the first author.

To that end, and based on the findings of the experiment, new hypotheses are being considered such as:

* A spreadsheet that contains range names in formulas will be more difficult to inspect and correct than a spreadsheet that does not use names in formulas.

Taking into account the limitations noted above, new research questions are under consideration relating to the hypothesis:

- Does the use of named ranges give the user (false) confidence in the accuracy of the spreadsheet?
- Does the use of named ranges increase the cognitive load on users debugging a spreadsheet?
- Do professional spreadsheet users face the same difficulties as novices when debugging a spreadsheet that contains named ranges?

In order to answer these new research questions, a new experiment is planned. This experiment will be based on the original experiment, but will be more carefully developed to ensure a balance between errors in named ranges, and errors in unnamed
ranges. The range errors will also be balanced between those where a wrong name is used, and those where the name refers to the wrong range. In order to answer RQ3 and RQ4, a more detailed analysis of the time each participant spends in each cell relating to the errors will be carried out. In order to answer RQ5, the experiment will be performed on both novice and professional spreadsheet users.

Based on the limitations found in the exploratory study it has been decided that computing students will not be used for this study, as their domain knowledge of variable names may have an affect on their use of range names. The group used in this study will be divided randomly into a control group and an experiment group. If possible, the T-CAT tool will be modified to record if the user makes use of the ‘Go To Name’ functions of the workbook.

A qualitative survey should be carried out on expert users to investigate their attitude to, and experience of, range names. An examination on the effect of naming ranges while developing a spreadsheet should also be undertaken, however no plans have been made regarding this yet.

6 CONCLUSION

While conventional wisdom might suggest that named ranges have a positive effect on spreadsheets, this work suggests otherwise, as the results indicate that named ranges may lead to a reduction in debugging performance of novice spreadsheet users.

These results do not, however, prove that named ranges will always have a negative effect on spreadsheets. Indeed it may have been the overuse of names, or the lengthening of formulas, that resulted in the poor performance by the experiment group, as the quantity of names used may well have distracted from their intrinsic worth. These results indicate that extreme care must be taken when advising users about developing spreadsheets. Development practices that might seem advantageous in principle can have unexpected consequences.

In essence, this paper supports, through a quantitative study, the possibility that range names may have a detrimental impact on spreadsheet debugging. Furthermore, it stresses the need for a larger and more tightly controlled and focused empirical study to answer this important question.

REFERENCES


This page intentionally blank
ABSTRACT

Whilst not all spreadsheet defects are structural in nature, poor layout choices can compromise spreadsheet quality. These defects may be avoided at the development stage by some simple mistake prevention and detection devices. Poka-Yoke (Japanese for Mistake Proofing), which owes its genesis to the Toyota Production System (the standard for manufacturing excellence throughout the world) offers some principles that may be applied to reducing spreadsheet defects. In this paper we examine spreadsheet structure and how it can lead to defects and illustrate some basic spreadsheet Poka-Yokes to reduce them. These include guidelines on how to arrange areas of cells so that whole rows and columns can be inserted anywhere without causing errors, and rules for when to use relative and absolute references with respect to what type of area is being referred to.

1 INTRODUCTION

The unstructured nature of a spreadsheet, allowing any layout and any type of formula, is a contributing factor to spreadsheet defects. There are no generally accepted good practice guidelines for spreadsheet development. Moreover, defects are often found long after the mistake has been made. Defect reduction is usually done by time consuming manual inspection (audit).

Poka-Yoke (pronounced "POH-kah YOH-kay") or mistake proofing, was first identified by Shigeo Shingo [Shingo, 1986] in the 1960s when, as a statistical process control engineer, he became frustrated that he could not achieve zero defects in manufacturing processes. Shingo realised that there was a clear distinction to be made between a mistake and a defect. A mistake was something that would inevitably lead to a defect unless one had a method to prevent or detect it within the manufacturing process. He also realised that “mistakes” were not always the fault of the operator particularly as the consequent defect becomes visible only at a later stage of the manufacturing process. The spreadsheet developer has little within Excel to help avoid defects and virtually nothing to aid good layout practice. In spreadsheets, defects can go undetected [EuSpRIG, 2007]. The goal of a Poka-Yoke device is to eliminate defects by preventing mistakes from occurring or at least detecting them at source rather than correct, at much greater cost, by audit when they become defects or errors (see Figure 1).

![Figure 1. The Poka-Yoke Advantage](image_url)
Well-known Poka-Yokes include the floppy-disk which can only be inserted one way into a drive. Software development, too, has its Quality Control methods [Rajalingham, 2000]. Indeed structured programming and object oriented design are, in a sense, prevention Poka-Yokes [Robinson, 1997].

In this paper we will describe defects due to poor layout and inadequate formula design and how they relate to Excel’s vulnerabilities. Structures and practices that can be employed to minimise defects during spreadsheet development and modification are shown. This includes the rules required for layout and the correct use of relative, absolute and mixed references. These rules, supported by our FormulaDataSleuth® (“Sleuth”) application [Bekenn, 2007], help to provide structure to a spreadsheet.

Moreover, the Sleuth implements SPYs (Spreadsheet Poka-Yokes). Both prevention and detection SPYs are described along with how they help developers to reduce spreadsheet defects. Some of these SPYs, e.g. good sheet layout and including an additional blank row/column at the bottom/right of a range, are encouraged by the use of the Sleuth but can be implemented without it. Other SPYs, e.g. grouping related areas and replication, are available as part of the Sleuth’s procedures.

1.1 Conventions

Some conventions used in this paper are:-

1. Data and formulas are not intermingled; they are always in separate blocks,
2. Spreadsheet structure and formula design is based upon Fill,
3. For the purposes of consistency and clarity we have adopted a convention that puts items in rows and instances in columns,
4. Data areas and cells are coloured Light Yellow in most examples,
5. Column letters and row numbers are shown top and left in most examples,
6. Some examples of referencing previous columns, inserting columns etc. are extendable to rows also.

1.2 Good Poka-Yokes (SPYs) Features

1. they are simple. If they are too complicated they will not be used,
2. they are part of the development process,
3. they are placed where the mistakes occur, to provide quick feedback.

2 STRUCTURAL DEFECTS

2.1 Mistakes

Human error has been studied in relation to spreadsheet mistakes [Panko, 2005]. However, errors can arise from defects particularly where modifications are being applied to existing workbooks. Such defects may arise from layout or formula mistakes. Some examples of mistakes and defects resulting in errors are listed below:-

1. Incorrect filling of formulas after modification due to the defect of the left column (or top row) formula being different to the rest,
2. Accidentally inserting rows/columns through unrelated areas of a spreadsheet not visible on screen due to defective layout,
3. Omitting to insert additional rows/columns in all “connected” areas throughout a workbook,
4. Failure to fill all areas of formulas after inserting rows/columns,

Care has to be taken when modifying workbooks and the flow of calculation needs to be understood. Irregular formulas and other defects have to be found so that filling from the top left and inserting rows/columns will not cause errors.

2.2 Excel Vulnerabilities

There are a number of types of error that occur due to Excel’s in-built vulnerabilities, which themselves may be considered as defects. Some examples are listed below:-

1. Range references e.g. “SUM(C8:C9)”, not adjusting when additional rows are inserted below the bottom of the range,
2. Formulas do not fill correctly due to incorrect use of relative, absolute and mixed references,
3. Incorrect formulas in a pasted area after copy and paste, due to clash of requirements for relative/absolute references needed for fill and copy.

3 SPREADSHEET STRUCTURE

Spreadsheet structure may be defined in terms of “Stripes”, “Groups” of Blocks within Stripes, and “Replication”, which are explained below.

It is also possible to conceive a Column or Row Insertion Test (CRIT) to test the resilience of a spreadsheet to modification. Any structure or formula which cannot have whole rows/columns inserted without causing errors is considered as having a CRIT defect.

3.1 “Single Cells”

“Single Cells” contain isolated items of data or one-off formulas and to fill requires an absolute reference, e.g. $C$12. Often Single Cells are constants used in a filled formula
area. Constants written into filled formulas would be unseen, difficult to audit and a potential source of defects. A SPY could be devised to detect this mistake, but a SPY could be obtrusive because some constants may be permissible (e.g. 2, 10, 100, and 1000).

3.2 “Blocks”

An area of data or filled formulas is referred to as a “Block” [O’Beirne, 2005]. Blocks are considered distinct from Single Cells. There are three types of Block:-

1. Single Row: Columns can be inserted within the Block.
2. Single Column: Rows can be inserted within the Block.
3. Multiple Row and Column: Columns and Rows can be inserted within the Block.

The absolute/relative row/column references (dollaring) that can be used in a filled formula must be chosen carefully (see section 10).

3.3 “Stripe”

A horizontal set of Blocks, spanning the same rows, make up a horizontal “Stripe”. A vertical set of Blocks, spanning the same columns, make up a vertical “Stripe”. Stripes are more important if they contain more than one row or more than one column.

3.4 “Idealised Layout”

An “Idealised Layout” is a sheet layout composed only of Stripes. These stripes can have whole rows or whole columns inserted at any point within a sheet, without introducing the risk of an error. Stripes are not allowed to overlap and must occupy their own set of rows/columns, but a row stripe can intersect a column stripe. Where possible, references within formulas are to cells on the same horizontal row or the same vertical column.

If selecting whole rows/columns slices through blocks of different sizes then it is likely that inserting rows/columns will result in a defect. The Sleuth contains a detection SPY for this so that the layout defect is avoided, see Figure 2.

3.5 “Calculation Chain”

A “Calculation Chain” starts from an input data item and finishes at a final result. Each non-data Block or Single Cell member of a Calculation Chain contains formulas that reference previous members of the Chain. Several Calculation Chains may exist in parallel giving different final results or they may combine at some point to form a single Calculation Chain.

3.6 “Group”

A horizontal or Row “Group” relates all the Blocks within horizontal Stripes of the same row height that are involved in a Calculation Chain. A vertical or Column “Group” relates all the Blocks within vertical Stripes of the same column width that are involved in a Calculation Chain. A Calculation Chain may flow through blocks of different heights/widths and thus several Groups can exist in the Chain. When rows/columns are to be inserted the insert must be repeated for all member Blocks of a Row/Column Group. The Sleuth contains a SPY to ensure that this happens by defining connected Groups, see figure 2.
4 CLASSES OF REFERENCE

Creation of defect-free formulas in a spreadsheet requires consideration of the different forms of reference.

4.1 Reference Type

The basic types of reference, “Relative”, “Absolute” and “Mixed” are defined in [O’Beirne, 2005]. A relative reference locates another cell with respect to the position of the cell containing the formula (un-dollared), e.g. “=C3” in cell G5 translates to “=R[-2]C[-4]”.

An absolute reference addresses a cell by its column and row address (dollared), e.g. “=$C$12” filled in cells G12 to I13 translates to “=R12C3”.

A mixed reference is absolute row and relative column or relative row and absolute column.

4.2 Entities Referred To by Single or Range References

A spreadsheet formula is a combination of references and mathematical operators calculating a result. The references can be any one or a mix of the above types. Moreover, these can be either a Single Cell or a filled Block. Figures (3) & (4) illustrate a range of possibilities.

For a single reference, e.g. =C3, =C7 or =$C$12:

Figure 3 Single References

The above situations are in order of potential ascending defect probability.

1. Any dollaring combination will work.
2. Reference must be relative (the default),
3. Reference must be absolute, (not the default), otherwise there is a defect leading to errors in the filled cells,
4. The formula has a CRIT defect because inserting columns will leave the formula to the right of the insert pointing to the wrong column; filling into the inserted blank cells only still leaves an error,
5. as for 4,
6. There is a CRIT defect, as inserting columns to the right of the block leaves the last column partial reference pointing to the wrong column and it may be unseen on another sheet.
For range reference =C3:D4 or =$C$3:$D$4:

1) Single Cell to Block
2) Block to Block
3) Self Reference (also Complex Reference)
4) Multiple Block Reference
5) Partial Reference to a Block
6) Partial Reference to Multiple Blocks

Figure 4 Range References

The above situations are in order of potential ascending defect probability. The blank rows/columns needed to avoid an obvious CRIT defect are omitted to simplify figure 4.

1. Any dollaring combination will work,
2. Reference must be relative column in C10:D10 and relative row in E8:E9,
3. Left half of complex reference must be absolute column (not the default),
4. The gap between the blocks is a defect as items placed in the gap, e.g. a temporary SUM(), cause an error as they are included in the column Q SUM(),
5. Some data may not be accessed anywhere due to the partial nature of the reference, the formula has a CRIT defect as inserting columns will cause errors,
6. as for 5 and the blocks must remain adjacent as the formula has a CRIT defect.

4.3 “Complex Reference”

A “Complex Reference” is a range reference where the start cell to the left of the colon is a different type of reference to the end cell to the right of the colon (see figure 5). When filled, the width or height of the range will be different in each cell. It can be used, for example, instead of previous column and self references for accumulating, thus removing the need for a Seed Value and still avoid a CRIT defect. Complex references are described in more detail in the Appendix (see section 10).

Figure 5 Complex References

5 GOOD LAYOUT AND FORMULA QUALITY

Spreadsheets are different to paper-based sets of calculations (e.g. accounts or journals). Spreadsheets can be easily modified whereas paper-based accounts or journals cannot be modified easily. However, it may be necessary to have a spreadsheet output presented in a similar form to accounts or journals but this does not mean underlying calculation process has to be laid out that way. A good spreadsheet layout is a) easy to understand and b) easy to modify. Under modification each action may have consequential actions, e.g. inserting rows requires re-filling of formulas. Ease of modification can be assessed.
by achieving a low number of consequential actions following each original action. Good Layout and Formula Quality are very closely related.

Consider a spreadsheet consisting of four quarters laid out in two vertical stripes (G:J & M:P) as shown in figure 6. The first block (G6:J7) contains the quantities in each quarter and the second block (M6:P7) comprises formulas for calculating the cumulative numbers from the first stripe. If all Single Row or Multiple Row and Column areas, in the second block, are to fill from the left hand column the formulas cannot have exceptions. However, the first column is “different” because, for it to fill, it needs to refer to the previous column (L). This is achieved by ensuring the previous column is available for a “Seed Value” set to “=0” (column L). A good layout practice is therefore to reserve an additional column to the left of a block (F & L in this case) for seed values.

An alternative in this case is to use a Complex Reference which does not require a Seed Value, in M6, “=SUM(SG6:G6)” filled to P6. A complex reference is more difficult to create correctly, but less defect prone.

Figure 6. Example of layout and formulas for Sections 5.1 to 5.5

5.1 Using Seed Values to ensure Blocks fill.
5.2 Stripe Based Idealised Layout

Figure 6 shows a Stripe based layout which is aimed at making insertion of rows and columns easy and error free (addressing the second mistake identified in section 2.1). As Stripes do not overlap, whole rows or columns can be inserted at any point without damaging the structure. A simple prevention SPY in the Sleuth using the Automatic Grouping feature allows this to happen. Also in an idealised layout references in any formula should be vertical or horizontal to the same column or row that contains the formula. Although the Percentage Increase formula in figure 6 shows an exception to this, in general, formulas which reference vertically or horizontally are easier to create and understand and require less use of absolute referencing (dollar characters).

5.3 Groups and Calculation Flow

Calculation within a spreadsheet often flows though a number of blocks of the same size progressively refining the calculations towards a final result. It is often the case that, additional rows (or columns) are required in all the Blocks within the calculation flow.

Excel has no direct procedure for ensuring that all the Blocks have the additional rows (or columns) entered. The developer has either to remember where all the Blocks in a calculation flow are or use some form of spreadsheet coding/annotation to remind them. This can lead to a spreadsheet defect from the mistakes 3 & 4 in section 2.1. The Sleuth application, allows the definition of a Group which comprises all the Blocks which are connected (see figure 2). This enables the possibility of inserting a row (or column) in one Block with the automatic insertion of a row (or column) in all the connected Blocks. Moreover, the Sleuth re-fills the formulas in all member Blocks of the Group. This prevention SPY (mentioned in figure 2) requires only that the user defines the connected Blocks using the same Group Name.

Whilst the traditional flow of calculation in a spreadsheet tends to follow the paper based equivalents, i.e. top left to bottom right, this has disadvantages when viewing part of a sheet on a screen rather than a complete two page ledger. It is better to keep the important Blocks which are to be viewed most often on the left hand side. The flow can then proceed from top left to the far right and then work through intermediate calculations from right to left finishing with the results to be viewed in the left hand columns. The final result is likely to end up at the bottom left of a sheet and can be linked back to the top left where it is the first thing to be viewed (as in figure 6). Also this final result at the top left is unlikely to move as a result of column or row insertion and is thus a safer item to link to another workbook.

5.4 Previous Column and Self References

The Percentage Increase formulas in figure 6 show a previous column reference to subtract the previous quarter from the Cumulative Quantity. This involves the Seed Value in column L. The Cumulative Quantity formulas again refer to the previous column but this is also a Self Reference as apart from column Q, which refers to the seed value, the other columns refer to the previous column within their own Block. Both Previous Column and Self References are higher risk as they do not take values from the whole Block being referenced. This leaves a chance that the last column of the block may not be accessed anywhere and may be defective, e.g. the last quarter may not carry forward to the first quarter of the next period. Also previous column references are damaged by inserting columns, a CRIT defect, and must be re-filled across the whole block to avoid some cases leading to errors.
5.5 Partial Access

If another year were added to figure 6, (see Figure 7), then the seed values for the cumulative quantities in column L would become a Partial Access to cells P6 and P7 to carry forward quarter 4. Such references are high risk due to a CRIT defect as they do not access the whole block and insertion of columns (say to convert figure 6 from Quarters to Months) may leave them in error pointing to the wrong column.

<table>
<thead>
<tr>
<th>G</th>
<th>H</th>
<th>I</th>
<th>J</th>
<th>L</th>
</tr>
</thead>
<tbody>
<tr>
<td>16</td>
<td>68</td>
<td>74</td>
<td>81</td>
<td>90</td>
</tr>
<tr>
<td>17</td>
<td>154</td>
<td>222</td>
<td>296</td>
<td>377</td>
</tr>
<tr>
<td>18</td>
<td>Item 1</td>
<td>Incremental Quantities</td>
<td></td>
<td></td>
</tr>
<tr>
<td>19</td>
<td>Item 2</td>
<td>Cumulative Quantities</td>
<td></td>
<td></td>
</tr>
<tr>
<td>20</td>
<td>83</td>
<td>125</td>
<td>173</td>
<td>226</td>
</tr>
<tr>
<td>21</td>
<td>287</td>
<td>369</td>
<td>454</td>
<td>549</td>
</tr>
</tbody>
</table>

Figure 7. Partial Access for Carry Forward

There are other cases where Blocks are referenced by a Partial Access, such as a rolling average over 3 months, and all are higher risk as some cells may end up not accessed and so not contributing to the final result. Also this type of partial access requires careful re-filling if columns are inserted as the formulas contain CRIT defects.

6 REPLICATION

Having created a set of calculations in a spreadsheet there is often a need to work on a second set of data, e.g. creating year 2 of a cash flow sheet from year 1. Copying the whole sheet to a new sheet is probably the safest way of achieving such a replication. If this second set is required below the original area, then Cut and Paste will move them from the temporary copied sheet. However, it is more convenient to Copy and Paste within a sheet, particularly when smaller areas of formulas are to be re-used. Unfortunately the formulas contain defects in how they will copy and paste but changing the dollaring would then cause defects with respect to fill. Copy and Paste or fill may convert these defects into spreadsheet errors. In fact, after pasting the new copy all the formulas require checking to ensure that combinations of relative and absolute referencing have produced the desired result in the pasted copy.

6.1 Apparent Adjustments Made by Copy and Paste

<table>
<thead>
<tr>
<th>B</th>
<th>C</th>
<th>D</th>
<th>E</th>
<th>F</th>
<th>G</th>
<th>H</th>
<th>I</th>
<th>J</th>
</tr>
</thead>
<tbody>
<tr>
<td>2</td>
<td>Quantities</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>3</td>
<td>Item 1</td>
<td>20</td>
<td>35</td>
<td>42</td>
<td>57</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>4</td>
<td>Item 2</td>
<td>10</td>
<td>15</td>
<td>22</td>
<td>36</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>5</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>6</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>7</td>
<td>Quantities Plus Percentage 1</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>8</td>
<td>Item 1</td>
<td>0.2</td>
<td>=G3*(1+$E$8)</td>
<td>=H3*(1+$E$8)</td>
<td>=I3*(1+$E$8)</td>
<td>=J3*(1+$E$8)</td>
<td></td>
<td></td>
</tr>
<tr>
<td>9</td>
<td>Item 2</td>
<td></td>
<td>=G4*(1+$E$8)</td>
<td>=H4*(1+$E$8)</td>
<td>=I4*(1+$E$8)</td>
<td>=J4*(1+$E$8)</td>
<td></td>
<td></td>
</tr>
<tr>
<td>10</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>11</td>
<td>Quantities Plus Percentage 2</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>12</td>
<td>Item 1</td>
<td>0.1</td>
<td>=G7*(1+$E$8)</td>
<td>=H7*(1+$E$8)</td>
<td>=I7*(1+$E$8)</td>
<td>=J7*(1+$E$8)</td>
<td></td>
<td></td>
</tr>
<tr>
<td>13</td>
<td>Item 2</td>
<td></td>
<td>=G8*(1+$E$8)</td>
<td>=H8*(1+$E$8)</td>
<td>=I8*(1+$E$8)</td>
<td>=J8*(1+$E$8)</td>
<td></td>
<td></td>
</tr>
<tr>
<td>14</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Figure 8. Typical Copy and Paste Problems

Copy and Paste does not adjust formulas (unless pasting to another workbook). It simply copies the R1C1 formula strings to a new set of cells. In many cases such as Blocks referencing Single Cell inputs the references will be absolute so that they fill correctly.
The pasted formulas will thus refer to the original copy of the Single Cell inputs rather than to the new pasted copies. The converse will also happen where a Block refers to a common Block of inputs which is not being duplicated, these relative references will end up pointing to a relative position on the sheet which has no relation to the inputs required. Both these possibilities are shown in figure 8, where the original formulas from G8:J9 have been copied and pasted to G12:J13 and a new percentage value entered in E12. The right hand section of Figure 8 shows that the formulas have not translated in the way the user must have intended. The user is now committed to a number of consequent actions which heighten the possibility of a mistake and a resulting error.

6.2 Replication as Intelligent Copy and Paste

To avoid these defects the Sleuth has a Replication procedure which allows the user to progressively mark all the Blocks and Single Cells which are to be duplicated, and then replicate all of them in one shot. The duplicated copies of the Blocks and Single Cells are placed in an unused area below the area involved in the replication and Cut and Paste can be used to move them to wherever they are required. In this way the number of consequent actions is reduced and possible errors from this cause are removed.

References are adjusted following two rules; (i) a reference to a cell being replicated points to the new copy of that cell and (ii) a reference to a cell not being replicated continues to point to it. These adjustments are done irrespective of the relative/absolute reference types. This requires that the dollaring has to be correct in the original so that Blocks are filled correctly. But this is no more than would have been required for the original to work correctly without replication.

Replication is a form of intelligent copy and paste working to reduce the number of consequent actions required by the user when they need to duplicate areas. This feature acts as prevention SPY device.

7 SPREADSHEET POKA YOKE

The Sleuth’s SPYs are summarised below together with the Mistakes, Defects and Errors that they address:-

1. Automatic fill of a Formula Block; prevents the use of different formulas in the left column / top row and encourages the use of Seed Values, ensures correct re-filling of blocks after changes to the formula,
2. Layout Defect Detection; warns when unrelated blocks of a different size are present in a Stripe and will prevent insertion of rows/columns if different groups overlap within a Stripe,
3. Connected Groups; removes the need for a developer to remember all the areas that interact and allows a one shot insert into all members of a Group with automatic re-filling of formula blocks,
4. Automatic re-filling of blocks when formulas are modified or rows/columns are inserted protects against errors caused by mistakes in re-filling or CRIT defects in formulas.
5. Automatic insertion of a blank row/column when inserting ensures that range references do not miss the additional rows/columns,
6. Precedents Reconciliation; detects Partial Accesses, mistakes in dollaring and references to cells outside of other blocks,
7. Formula Correction; adjusts the dollaring and the size of ranges to match the blocks being referenced,
8. Replication; ensures that references in the replicated areas are correct and removes the need for the checking of formulas that is needed after copy and paste,

8 CONCLUSION

This paper has examined spreadsheet structure and how “mistakes” can make a spreadsheet defective. These “mistakes” are sometimes due to Excel vulnerabilities not being appreciated, sometimes due to lack of forethought or awareness of how the spreadsheet might need to be modified. Drawing a parallel with the overwhelming success of notions of quality to reduce defects in manufacturing using the pioneering Toyota Production System we have suggested that Poka Yoke, i.e. mistake proofing, could be a valuable technique to reduce defects and therefore the risk of error.

Our detailed consideration of spreadsheet structure has lead to identifying a number of defects and some Spreadsheet Poka Yoke (SPYs) to prevent or detect the defects before they are propagated. Defect reduction at the development stage will potentially save time and effort on auditing. The application of a Column Row Insertion Test (CRIT) and the attendant SPYs in our FormulaDataSleuth® (Sleuth) application along with the Replication SPY has been tested in a real application for a complex cost-modelling scenario, which will be reported.

Although we have not considered the out-and-out human error aspect of “mistakes” it is our belief that some of these may yield to the SPY approach.

9 REFERENCES


Panko, R (2005), “What We Know About Spreadsheet Errors” http://panko.shidler.hawaii.edu/SSR/Mypapers/whatknow.htm 12:06pm 01/02/2008

O’Beirne, P (2005), Spreadsheet Check and Control: 47 key practices to detect and prevent errors, Systems Publishing ISBN 190540400X
Relative/Absolute Referencing (“Dollaring”) is the key to making Blocks fill correctly so that there is the same formula in each cell. It also affects how formulas Copy and Paste (but NOT how they Cut and Paste). A brief summary is included below in a matrix. Any deviation from the possibilities listed below will cause errors when filled. Where alternatives are indicated both alternatives will fill correctly but the result of using Copy and Paste will be different for each.

<table>
<thead>
<tr>
<th>Normal Reference relative / absolute requirements</th>
<th>Reference to single cell</th>
<th>Reference to single row Block</th>
<th>Reference to single column Block</th>
<th>Reference to multiple row and column Block</th>
</tr>
</thead>
<tbody>
<tr>
<td>Formula in a single cell (not filled)</td>
<td>Any reference</td>
<td>Any range</td>
<td>Any range</td>
<td>Any range</td>
</tr>
<tr>
<td>Formula filled across a single row</td>
<td>Absolute column any row</td>
<td>Relative column any row</td>
<td>Absolute column any row range</td>
<td>Relative column any row range</td>
</tr>
<tr>
<td></td>
<td>($G5$ or $G$5$)</td>
<td>($G5$ or $G$5$)</td>
<td>($G5$5:$G$5$8$ or $G$5$5:$G$5$8$)</td>
<td>($G5$8:$G$5$8$ or $G$5$5:$J$5$8$)</td>
</tr>
<tr>
<td>Formula filled down a single column</td>
<td>Any column absolute row</td>
<td>Any column absolute row</td>
<td>Any column relative row</td>
<td>Any column relative row</td>
</tr>
<tr>
<td></td>
<td>($G5$ or $G$5$)</td>
<td>($G5$5:$G$5$5$ or $G$5$5:$J$5$5$)</td>
<td>($G5$ or $G$5$)</td>
<td>($G5$5:$G$5$5$ or $G$5$5:$J$5$5$)</td>
</tr>
<tr>
<td>Formula filled across and down into multiple columns and rows</td>
<td>Absolute column absolute row ($G5$5)</td>
<td>Relative column absolute row ($G5$5)</td>
<td>Absolute column relative row ($G5$5)</td>
<td>Relative column relative row ($G5$5)</td>
</tr>
</tbody>
</table>

The matrix below shows the Complex Reference version of range references. The SUM() function is used in the examples to generate cumulative values. Some other functions such as COLUMNS() or ROWS() can be used in the same way. Careful use of Complex References can provide alternatives to formulas that have CRIT defects (see example below).

<table>
<thead>
<tr>
<th>Complex Reference relative / absolute requirements</th>
<th>Cumulative Sum of single row Block</th>
<th>Cumulative Sum of single column Block</th>
<th>Cumulative Sum of all rows of multiple row and column Block</th>
<th>Cumulative Sum of all columns of multiple row and column Block</th>
</tr>
</thead>
<tbody>
<tr>
<td>Formula filled across a single row</td>
<td>Complex column any row range</td>
<td>Not applicable</td>
<td>Not recommended SUM($G5$5:$G$5$5$) or SUM($G$5$5:$G$5$5$)</td>
<td>Not applicable</td>
</tr>
<tr>
<td>Formula filled down a single column</td>
<td>Not applicable</td>
<td>Any column complex row range</td>
<td>Not applicable</td>
<td>Not recommended SUM($G5$5:$J$5$5$) or SUM($G$5$5:$J$5$5$)</td>
</tr>
<tr>
<td>Formula filled across and down into multiple columns and rows</td>
<td>Not applicable</td>
<td>Not applicable</td>
<td>Complex column relative row range SUM($G5$5:$G$5$5$)</td>
<td>Relative column complex row range SUM($G5$5:$G$5$5$)</td>
</tr>
</tbody>
</table>

The following tables demonstrate the use of Complex References:

<table>
<thead>
<tr>
<th>C</th>
<th>D</th>
<th>E</th>
<th>F</th>
<th>G</th>
</tr>
</thead>
<tbody>
<tr>
<td>3</td>
<td>-20</td>
<td>05</td>
<td>85</td>
<td>F3-E3</td>
</tr>
<tr>
<td>4</td>
<td>F45-G55</td>
<td>=E4-E5</td>
<td>=F3-E3</td>
<td>Incremental</td>
</tr>
<tr>
<td>5</td>
<td>1) Incremental with Seed Value</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>6</td>
<td>0</td>
<td>05</td>
<td>85</td>
<td></td>
</tr>
<tr>
<td>7</td>
<td>INDEX($C7:$F7;COLUMNS($C7:$E7);$INDEX($C7:$F7;COLUMNS($C7:$F7);1))</td>
<td>=INDEX($C7:$F7;COLUMNS($C7:$F7);1)</td>
<td>Incremental</td>
<td></td>
</tr>
<tr>
<td>8</td>
<td>2) Incremental with Complex Reference</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>9</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>10</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
Embedded Spreadsheet Modelling

Angela Collins,
BDO Stoy Hayward LLP
3 Hardman Square, Manchester, M3 3EB
angela.collins@bdo.co.uk

ABSTRACT

In larger accounting firms, specialist modellers typically sit in separate teams. This paper will look at the advantages of embedding a specialist modeller within a Corporate Finance Team.

1. Introduction

This paper looks at the role of a modelling specialist based within the Corporate Finance (CF) Department of BDO Stoy Hayward LLP in Manchester. All other modellers also sit within Corporate Finance but are based together in London. It appears to be a feature of most of the larger accounting firms that modellers are based in separate teams rather than being distributed throughout the business. Modelling seems to be viewed in a similar way to Audit – not something which can be done alone. Due to this separation it can be difficult to improve the modelling skills of non specialist staff.

2. Some issues

2.1 Image

One of the issues which modellers face is the misconception that they are all technical specialists who do not understand the commercial realities under which most Corporate Financiers operate. There is the feeling that it is not possible to follow best practice due to time constraints. (An even more common issue is the lack of understanding of the importance of ‘best practice’.)

2.2 What is ‘best practice’?

It can be difficult to persuade non-specialists of the importance of ‘best practice’ especially as there is no definite agreement amongst modellers about what constitutes ‘best practice’.

It seemed appropriate to focus on a small number of points at first so that non-modellers did not feel overwhelmed and therefore discouraged. As these points become embedded and their advantages seen then future points may be more easily accepted.

These basic points were - 1 – No circular references, 2 – No hard-coding of values within formula, 3 – Test as you go along, 4 – Formulae to be consistent. 5 – Separate inputs, calculations and outputs – make it clear which of these each cell is. 6 – Keep it simple!

3. Corporate Finance Survey
3.1 Background

A survey was conducted (via email) in BDO Corporate Finance in Manchester. Participants were advised that a paper was being prepared which would look at the role of a modelling specialist embedded in CF for EuSpRIG and they were asked the following questions:
1) Have you found it useful having a modelling specialist based here?
2) Is there anything in particular that has helped?
3) Is there anything you would like done differently?
4) Have you changed the way you work with spreadsheets as a result of working with a modelling specialist?

3.2 Responses

Responses were received from 11 people from a team of 28.

Q1 – Have you found it useful having a modelling specialist based here?

All respondents agreed that having a modeller integrated into CF was useful. The benefits include improved risk management, the ability to sell modelling as a stand-alone service, the ability to take on more complex modelling assignments and improved credibility with external institutions.

Q2 – Is there anything in particular that has helped?

The delivery of in-house training which has ‘dramatically improved’ Excel skills within the department. Availability of a named person as a trouble-shooter. Improved ability to resolve technical issues in-house. Educating CF about tools in Excel which have significantly cut down on time spent on financial analysis.

Q3 – Is there anything you would like done differently?

As there has been a growing understanding of the benefits that a specialist modeller offers there have been increasing requests for modelling support. This can be difficult to schedule when there is only one modeller in the office.

Q4 – Have you changed the way you work with spreadsheets as the result of working with a modelling specialist?

This included two main areas, 1 – Specific techniques (including how to use flags and how to used named ranges in calculations. 2 – Best Practice - this referred to in a number of responses. Particularly mentioned were improved version control and not hard-coding values into formulae.

3.3 Isolation

Issues
One of the issues that must be addressed if you work on your own as a modeller is that of isolation. There are no immediate colleagues performing a similar role and this can be difficult, especially when there is a technical issue to resolve. It has been useful to build a strong relationship with other modellers within BDO. Frequent modelling updates are held and work is spread around the team where possible.

4. Training

4.1 Formal

There is an existing in-house modelling course which takes one day at the basic level but can run over two days, (this would include half a day examining how to review a model). It can be difficult for people to commit to two days, especially if they are concerned about their existing level of skills.

The course is currently being developed on a modular basis. This will enable us to tailor courses to a variety of audiences and it is hoped that this will increase attendance.

4.2 Ad-hoc

A-hoc training has been popular. Especially if staff have a problem when building a model. It is possible to fix the model and send it back and this would be the quickest solution at the time. However next time the issue arose the staff member would not be able to resolve it and the modeller would need to intervene again. It is more useful to sit with the person involved and show them one way of fixing the issue. Where possible alternative solutions that may be used in different circumstances are demonstrated. Training delivered in segments like this can be more easily absorbed and is seen to be directly relevant to the role that person performs (McQuire, 2007). There are some excellent external training courses available but a course is only the beginning. On-going support is crucial in developing modelling skills.

Although the general level of skills in the department has improved it is unrealistic to expect that everyone will develop into a modelling expert. The main aim is that people learn what not to do and when they need to ask for help. Advanced Excel users can sometime forget what it is like to not know the basics. This became clear when a short course was presented to another department. Rather than the course content being dictated by what XXX the attendees were asked what they wanted to learn. Their responses included how to use shortcut keys, how to merge data from two columns into one and how to work with multiple sheets. Feedback after the course showed that there were two elements which were found to be particularly useful, one was general best practice and the other was the use of Edit – Replace. This is a function which most modellers will use without thinking but, as the course was running, it became clear that this was new to the attendees and it was therefore explained in detail. This flexibility and awareness has proved crucial. It is more effective to have simple techniques understood than to have advanced techniques only half absorbed.

5. Assumptions/Model Review/Testing/Documentation

When model are developed internally there is a reluctance to produce a specification, and those specifications which are produced are generally inadequate (Pryor 2003).
Many modellers have had the experience of building a model which implements all the assumptions supplied. A sophisticated, well constructed and robust model is presented to the client. They look at the results and believe that the model is wrong. What they actually mean is they don’t like the results. It is therefore crucial that all conversations regarding a model are documented and that you can demonstrate to the client how their assumptions have been implemented.

Providing full documentation is perceived to be onerous and something which can be allowed to slip when under time-pressure. Once simple way to increase ease of documentation is the use of cell comments. A simple macro can be run which will extract all the cell comments from a model and this can form a basic documentation.

Users are encouraged to perform a substantial amount of testing. (Pryor, 2004 – ‘Testing is the only way to tell what the spreadsheet actually does’). One technique which is popular is to use unrealistic inputs. For example it is unlikely that fixed assets will depreciate over 2 months. It is however a useful way of checking that your straight line depreciation works and doesn’t over depreciate. It is also worth deleting all your inputs (providing you have made a back-up copy of your model first)! Then run the model with only single sections of inputs complete and check that the results are as expected. One of the errors frequently found is that cashflows occur in the wrong period. A quick way to test this is to use excessive inputs. i.e. if there is a fixed asset purchase of £1,000 add a lot of noughts to the end. When you look at the Financial Statements it will be clear if the timing is correct.

6. Developments

6.1 Template models

It is tempting to pick up a model which has been developed for a previous job and amend that. This poses a number of risks: the model may only be appropriate in some circumstances. If you pick up a model which someone else developed you may not understand how it works. Also, if a model has been used previously there is a tendency to assume that it works and there is a reluctance to invest further time & cost in testing.

Principles of template models

There needs to be a consensus about the content of a template model. They should not be developed by one modeller who works in isolation. When a model is built for an external client a model specification will (should) be approved. – the same principle applies to template models built for use within the business. The basic template is controlled by an identified member of staff to ensure that any development is appropriate and documented.

It is key that the limitations of a template model are understood. It is unlikely to cover every circumstance that may arise. When a template model is picked up for use on a new job one approach is to ensure that all inputs have been cleared. This prevents existing inputs being used in error and not updated when appropriate.

In common with other firms in the sector BDO have a base model which is now used for all PFI jobs. Substantial time was spent in developing the model to ensure that it would meet standard specifications as these are generally well defined (Croll 2005). When the first draft version was available a small group of PFI specialists gathered together and worked through the model, looking at all inputs and examining how these were taken through the calculations.
As the model is used in a variety of jobs it is further refined. As these refinements arise the PFI team consider if they are specific and should therefore be confined solely to the job on which they arise. If they are more generic changes then the base template model is updated and, where appropriate, these changes are applied to any other current jobs.

6.2 Code library

There are a number of elements which are common to many models developed. The modelling team have recently developed a code library which will contain adaptable examples of code for these common elements such as depreciation, working capital etc. It is believed that this modular approach will improve the accuracy and speed of model builds.(Paine 2007).

6.3 Modelling Road-shows

The approach discussed has proved useful in the Manchester office. The core modelling team are keen to develop links between ‘official’ modellers and other offices. A leaflet has been developed and is being distributed to all offices. On one side of the sheet there are some hints and tips, shortcuts etc. The other side describes the services offered by the modelling team and contact details. Once this leaflet has been distributed to an office an informal modelling road-show will be held which will cover some best practice principles and encourage engagement with the modelling team.

7. Conclusion

Modelling experts do not operate in isolation. They work so that clients, both internal and external, can analyse and understand information and make appropriate decisions. We deliver the best product when we understand the environment in which these clients operate

8. References


NLP-SIR: A Natural Language Approach for Spreadsheet Information Retrieval

Derek Flood, Kevin Mc Daid, Fergal Mc Caffery

Dundalk Institute of Technology

derek.flood@dkit.ie, kevin.mcdaid@dkit.ie, fergal.mccaffery@dkit.ie

ABSTRACT

Spreadsheets are a ubiquitous software tool, used for a wide variety of tasks such as financial modelling, statistical analysis and inventory management. Extracting meaningful information from such data can be a difficult task, especially for novice users unfamiliar with the advanced data processing features of many spreadsheet applications. We believe that through the use of Natural Language Processing (NLP) techniques this task can be made considerably easier. This paper introduces NLP-SIR, a Natural language interface for spreadsheet information retrieval. The results of a recent evaluation which compared NLP-SIR with existing Information retrieval tools are also outlined. This evaluation has shown that NLP-SIR is a more effective method of spreadsheet information retrieval.

1. INTRODUCTION

The latest version of Microsoft© Excel can contain over 1 million rows per worksheet facilitating the storage of considerable amounts of data. With such a large volume of data, extracting meaningful information can be a difficult task. Features such as Filters and PivotTables can assist. However, for novice spreadsheet users the use of these features can cause great difficulty.

The aim of this research is to investigate and improve the effectiveness and efficiency of spreadsheet information retrieval. One way this improvement can be achieved is through a Natural Language Interface. Natural Language Interfaces, or NLI’s, allow a user to operate a computer application by telling the computer what they want, in their own way. The application then interprets this, through the use of NLP techniques, and performs the desired action on behalf of the user.

NLI’s have been demonstrated to improve the efficiency with which users interact with applications. Chart generation [Kato, Matsushita et al., 2002] and accessing databases [Liddy and Liddy, 2001] are just some of the domains which have benefited from the use of NLI’s. There are many ways in which NLP may be performed, some of which are outlined in section 2.

NLP-SIR is a NLI for spreadsheet information retrieval. The system allows users to perform common information retrieval tasks, such as filtering and generating summary tables, similar to PivotTables, through the use of natural language. Section 3 details the
NLP-SIR system and also highlights some of the limitations of the current implementation of the system.

The effectiveness and efficiency of the NLP-SIR system were evaluated through a controlled experiment. This experiment compared NLP-SIR with existing information retrieval tools found within Microsoft© Excel. A full description of this evaluation along with the results can be found in Section 4. Section 5 outlines some ways in which this technology can be improved. Section 6 then concludes this paper.

2. BACKGROUND

Spreadsheets are a versatile software application. They allow complex simulations to be run quickly and different scenarios to be modelled with little modification. All of these tasks are accomplished through the use of formulae, however a large number of spreadsheets contain no formulae at all. These spreadsheets are used purely to store data which can be later analysed for additional information.

The Euses spreadsheet corpus[Fisher and Rothermel, 2005] is a collection of over 4,000 real-world spreadsheets, which have been collected from a variety of sources primarily the World Wide Web. An inspection of this corpus, by the author, has shown that the three largest spreadsheets, each containing over 1 million cells, contain no formulae. These spreadsheets instead contain static data which can be interrogated for more detailed information.

Spreadsheet tools such as PivotTables and Filters can be used to help analyse this data, however for novice spreadsheet users, these features can be difficult to use effectively. Although more experienced users find using these features to be straightforward, they still require multiple interactions, which can be time consuming.

There are many ways in which complex features like these can be simplified. One approach is the use of a command based interface. In previous work [Flood and McDaid, 2007] the authors have developed a command based navigation system to simplify the way in which users of voice recognition technology navigate a worksheet. This approach allowed the user to perform certain navigational tasks, which would have taken multiple interactions with existing voice recognition technology, more efficiently.

Another approach that may simplify these tasks is through NLP. NLP is a way for computer applications to interpret human language. By using a NLI a user can operate a computer by telling the computer what they want in their own way. The input from these interactions is often referred to as an utterance. A number of applications[Zelle and Mooney R. J., 1996; Liddy and Liddy, 2001; Tang and Mooney, 2001; Kato, Matsushita et al., 2002; Begel, 2005; Woodley, Tannier et al., 2006], including chart generation, have benefitted from NLI’s.

A common way to analyse large amounts of data is through charts. Charts can also be used to highlight characteristics of data that would be otherwise indistinguishable. As these characteristics become visible, they will lead the user to ask further questions which can then be answered through additional charts. Kato et al [Kato, Matsushita et al., 2002] propose the use of a natural language interface to help users in this type of exploratory data analysis.

Kato’s system allows a user to express, in natural language, what they would like to have presented in a chart. When a chart has been generated the user can then alter it through...
additional utterances to further interrogate the data. For example, if a user was looking at
sales data for Ireland over the last ten years and wanted to see a visual representation of
just those sales made in County Louth in 2004 and 2005, they might say “Show me the
sales in Louth for 2004 and 2005”. The system would then respond with a chart showing
the sales in Louth for those 2 years. After seeing this chart the user may wish to see this
same information broken down by town and might say “By town”, which would cause the
system to alter the chart to show the sales in Louth for 2004 and 2005 broken down by
town.

In using this approach, the user does not need to go through the steps of manually
creating each chart. Instead the system can automatically determine the right type of chart
and the correct data to include before generating the correct chart. The system can also
determine changes to the parameters when subsequent utterances are given. This
approach has the added benefit of enabling inexperienced users to generate charts quickly
and with minimum effort.

In order to evaluate the effectiveness of the system, Kato et al[Mitsunori, Eisaku et al.,
2004], asked 25 participants to select the chart, from multiple possibilities, which most
accurately depicted the information required by a natural language statement. The
participants and the system were presented with six different statements and in 4 of these
scenarios participants selected the same chart as the system generated. In the remaining
cases the chart generated by the system was the second most popular.

Reducing the number of interactions is not the only way in which NLI’s can improve
users’ effectiveness. Structured Query Language (SQL) may be used by users to retrieve
information from a database. In order to use SQL effectively however users need to know
the structure of the database as well as the syntax of the SQL language. This scenario is
not ideal for users who have no background in database technologies.

In order to facilitate the processing of queries from users with no statistical background
but who needed access to statistical information, Liddy et al[Liddy and Liddy, 2001] have
developed a natural language based system. This system uses NLP techniques to interpret
the information that is being sought from a natural language style query. By testing the
system on a small sample of user queries, Liddy et al found that 95% of the new user
queries could be covered and covered accurately by their system.

3. NLP-SIR

3.1. System Overview

When the concept of a natural language interface to spreadsheets was first conceived a
number of characteristics of such a system were immediately apparent. As a wide variety
of spreadsheets are used in different situations it became evident that a NLI for
spreadsheet information retrieval must be adaptable and could not be tied to any one
domain. A number of NLI’s, especially those in the database domain, for example[Zelle
and Mooney R. J., 1996], are constructed for a particular database. Therefore, applying
the same system to another domain would not be successful without some modification or
additional training.

Another characteristic that is essential to a NLI for spreadsheet information retrieval is
flexibility. In order to accommodate natural language any system should be able to
identify the meaning of an utterance independent of the structure of that utterance. For
example consider a spreadsheet containing a list of golf courses situated in the state of
Indiana. Such a spreadsheet could contain information such as the number of holes on the course, the type of terrain and the price of playing on the course. While looking at this spreadsheet, one user might say “what golf courses in Marion have executive difficulty” Where as another user might ask for a “list of golf courses that are executive and in Marion”. Both users are looking for the same information but the way in which they ask is structured differently.

The result of incorporating these characteristics into a NLI for spreadsheets is NLP-SIR, (Natural Language Processing for Spreadsheet Information Retrieval). This system allows a user to perform certain information extraction functions by telling the system what they would like in their own way. The present system allows the user to filter out rows or columns, to count the number of rows that meet a certain set of criteria and to find the most or least frequent value in a given column. It also allows users to generate tables, similar to PivotTables which count the number of rows that meet certain criteria.

The current implementation of the system uses a text based interface where a user types in what they would like to ask the system. The answer generated by the system is presented to the user through an alert box, similar to that shown in Figure 1.

![Figure 1: NLP-SIR](image)

The spreadsheet depicted in Figure 1 shows a list of golf courses that are situated in the state of Indiana. There are over 120 courses in total and extracting meaningful information, such as the number of courses of each difficulty, from this data can be time consuming. To find this information using the conventional interface, a user would need to create a PivotTable doing a count of each difficulty type. With NLP-SIR the same user could type in the question “How many courses are of each difficulty?” and the system would find the information and present it to them, eliminating the need for them to manually create a PivotTable.

During the course of developing NLP-SIR, a number of natural language systems were evaluated to see if any of these existing approaches could be used within the spreadsheet domain. Domains such as databases [Zelle and Mooney R. J., 1996], Chart generation [Kato, Matsushita et al., 2002] and XML retrieval [Woodley, Tannier et al., 2006] were all examined and each system offered a unique approach to the problem. It was found however that none of these systems contained all of the desired characteristics. The
approach used by NLP-SIR to infer meaning from the utterance has similarities to the systems presented above.

3.2. System Assumptions and Limitations

The present implementation makes certain assumptions about the spreadsheet. It assumes that the data is well structured and that each row in the spreadsheet corresponds to one entry. The system also assumes that the first row within the spreadsheet contains the name of each column. As well as this it is assumed that only one data type is featured within each column. The spreadsheet structure supported by NLP-SIR is a common structure for storing data.

In addition to the above assumptions NLP-SIR is limited in the utterances it will correctly recognise. When users are referring to a column, they must use the column name as it appears in the first row of the spreadsheet. For example if a user wanted to refer to the Price column in the above spreadsheet they would need to use the word “Price” and not other synonyms, such as cost. In certain instances whitespace would appear within column headings. The NLP-SIR system ignores whitespace and looks only at the characters that appear within the heading.

When a user wants to refer to a numeric column the user must explicitly name this column. With textual columns the user can use the text as it appears within the spreadsheet without the need to state the name of the column which contains the text. For example if a user wanted to see all of the courses with an easy difficulty they could say “Show me all of the easy courses”. The system can automatically recognise the column in which the value “easy” appears.

If a value such as this appears in many columns, the system can ask the user for clarification before applying the desired function. The system will alert the user to the columns the data element appears in while asking for clarification. This way the user can see exactly what the system did not understand.

The present implementation of the system is based on positive statements, i.e. the user instructs the system as to what they want. In some circumstances it may be desirable to tell the system what they don’t want, for example if a user wanted to see all of the golf courses that did not have a varied terrain. This type of query is not implemented in the current system; however this will be included in future versions of NLP-SIR.

Another limitation of the system is the output mode. When the system displays additional information, such as pivot tables they are displayed in an alert box. This is not an ideal approach as it causes users difficulty in evaluating the systems results. If the results are quite large then the alert box exceeds the screen size and the user is unable to see part of the results.

4. EVALUATION

In order to test the effectiveness of the NLP-SIR system, a comparative study was conducted. The initial evaluation focused on novice spreadsheet users as it is believed that this group of users would benefit most from this new technology. It is hypothesised that novice spreadsheet users would be able to perform information retrieval tasks, more effectively and more efficiently using NLP-SIR than through using the existing Microsoft© Excel interface.
4.1. Experiment design

In order to test this hypothesis, 41 participants were asked to take part in a comparative experiment. The participants selected were 4th year students studying for a Bachelor of Business Studies (Honours). One of the modules that the participants had taken was on spreadsheets and in particular Microsoft© Excel. Although the participants had used Microsoft© Excel in previous years, it is believed that, for most participants, this year was the first time that they used the 2007 version including the updated interface.

As part of their spreadsheet course, the participants were divided into four groups in order to ensure adequate facilities were available for each student. Each group had a one hour lab class per week and it was during these lab classes that the experiment was conducted. All participants were given a brief description of the study.

Once the participants had agreed to take part, they were given a brief overview of the experiment. Each class group was randomly divided into two groups, Group 1 and Group 2. Group 1 were asked to complete the tasks using the existing technology while group 2 were asked to use the NLP-SIR system. Before they began the task, the participants were given a brief recap on how to use Filters and PivotTables on a sample spreadsheet. They were also shown how to perform the same tasks using the NLP-SIR system.

For the evaluation, participants were asked to retrieve information from a given spreadsheet. This spreadsheet contained a list of 127 golf courses within the state of Indiana. Information such as the type of terrain, the number of holes and the difficulty of each course was detailed in this spreadsheet. The participants were told that they were employed in a tourist office and were asked to write down the information that would satisfy some requests for information that had been received. These requests, similar to R1, asked for either a list of the courses that met a certain set of criteria or the number of courses that met these criteria.

R1: Hi, I am a novice golfer and I am going on holidays to Hancock. While there I would like to try some of the easy courses. Can you provide me with a list of such courses with a varied terrain?

The requests for information were displayed to the participants in a task pane similar to that in Figure 2. When the participant had found the information they needed and written it down, they were required to press the “next task” button to move on to the next task. When they had completed the last task a “thank you” message was displayed.

In total 10 tasks were given to each participant. These tasks were divided into three levels of difficulty; Easy, Intermediate and Complex. The tasks were divided based on how difficult it would be to accomplish them using the existing technology. The first 4 tasks were classified as easy and required participants to filter the data, based on a set of criteria, such as having a terrain of flat and a difficulty of easy, or to find the most common type of terrain. The next 3 tasks were classified as intermediate and asked participants to create a PivotTable with a varying number of filters and columns. One of these tasks asked participants to find the number of executive courses in each county. The final 3 tasks were classified as complex and required participants to perform an “or” type query over multiple columns. One such task asked the participants to find the number of courses that either had a hilly terrain or had a difficulty level of hard.
No time limit was set for the task, allowing users to finish when they felt they had completed all of the tasks. The majority of participants had attempted all of the tasks however some participants only attempted the first few tasks. When the participants had finished the experiment they were asked to complete a short questionnaire regarding their experience of the technology and of the tasks. The aim of this questionnaire was to evaluate the perceived experience of using the technologies in a given situation.

4.2. Results

While the participants were doing the task, a customised macro was run in the background to record the time of each cell change and each cell selection, the time and details of each filter that was applied to the spreadsheet and finally the utterances that were given to and results that were received from the NLP-SIR system. This information was used as a basis for evaluating the effectiveness and efficiency of the system.

The effectiveness of the system was measured using the number of tasks that the participants had completed successfully and the number of utterances that were required for each task. The efficiency of the system was evaluated using the total time that it had taken participants to complete each task.

The recording macro failed to record data for 3 participants resulting in data for just 38 participants. 20 of these participants had used Microsoft© Excel for the task while the remaining 18 had used the NLP-SIR system.

Successfully Completed Tasks

During the experiment participants were asked to write down the information that would answer the requests that they had been given. The first three tasks required users to write down the name of the golf courses that would satisfy a given set of criteria. The fourth task then asked participants for the most popular type of terrain. The remaining tasks asked users to write down the number of courses that met a certain set of criteria.

Each participant’s written answers were manually reviewed by the author to determine if the answers they supplied matched the pre-determined answers. In some cases the authors’ judgement was used to determine if the supplied answer was in fact correct, for
example some participants wrote down the address of the course instead of the name, in this particular scenario the answer was deemed to be correct. Only a small number of exceptional cases arose and were seen in both sets of results. In each case the judgements made were consistent.

Table 1: Average number of tasks completed successfully

<table>
<thead>
<tr>
<th></th>
<th>Excel</th>
<th>NLP-SIR</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Total Number of Tasks</strong></td>
<td>3.43</td>
<td>6.75</td>
</tr>
</tbody>
</table>

Table 1 shows the average number of tasks that were successfully completed by participants. On average the participants who were using the NLP-SIR system were able to complete 6.75 of the 10 tasks correctly. The participants who used the Microsoft© Excel technology however were only able to correctly complete, on average, 3.43 of the tasks. Using non-parametric methods the statistical significance of these results were measured and a p-value of .0003 was found indicating that the difference in performance is statistically significant. In order to investigate further where the NLP-SIR system was of most benefit to participants, the number of tasks completed successfully in each category was also examined.

Table 2: Average number of tasks completed successfully by category (*P-value given to 2 places of decimal)

<table>
<thead>
<tr>
<th></th>
<th>Easy</th>
<th>Intermediate</th>
<th>Complex</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Excel</strong></td>
<td>2.85</td>
<td>1.10</td>
<td>0.05</td>
</tr>
<tr>
<td><strong>NLP-SIR</strong></td>
<td>3.22</td>
<td>1.06</td>
<td>1.78</td>
</tr>
<tr>
<td><strong>P-Value</strong></td>
<td>.13*</td>
<td></td>
<td>.00*</td>
</tr>
<tr>
<td><strong>Total Number of Tasks</strong></td>
<td>4</td>
<td>3</td>
<td>3</td>
</tr>
</tbody>
</table>

Table 2 shows the average number of tasks that were completed successfully for each category of task. This table shows that participants who were using the NLP-SIR technology performed better while doing the Easy and Complex tasks; however it was found that the opposite occurred with the intermediate tasks.

In order to determine why this was the case, the utterances that were used by the participants of the NLP-SIR system were examined. It was observed that some of the utterances that were given to the system used phrases, such as “each of the counties”, that were given in the scenario description. In the case of Task 5 and Task 7, both of which are of intermediate difficulty, these phrases were not recognised by the system and the answer that was given to the participants was incorrect. It is planned to investigate this issue further to determine the extent with which the phraseology of the scenarios influenced the participants’ choice of language.

It should also be noted that only one of the participants using Microsoft© Excel was able to complete any of the complex tasks. This participant correctly answered one of these tasks while the participants who had used the NLP-SIR system were able to successfully complete on average 1.78.

**Average time to complete each task**

The time at which each user clicked on the “Next Task” button was recorded during the experiment. This time was used to determine how long each participant spent on each
task. The task time was calculated by taking the time each participant started the task away from the time they moved to the next task. If the task pane was closed during the trial, participants would have had to return to the beginning and move through each of tasks again until they reached the task they were on prior to its closure.

While the authors were introducing the technology, some of the participants opened the spreadsheet and clicked onto the first task. They then waited until instructed before beginning the task. This has meant that in some cases the recorded time was not a true reflection of how long it took to complete the first task as it included the time spent by participants listening to the instructions that were being given.

When participants reached the tenth and final task, some of the participants failed to click through to the “thank you” message. This has meant that the time to complete the final task is not a true reflection of the actual time it took the participants to complete the task.

For these reasons the first and final tasks have not been included in the averages presented here. As these two tasks have been discarded, the average times presented here are based on three easy tasks, three intermediate tasks and two complex tasks.

<table>
<thead>
<tr>
<th></th>
<th>Easy</th>
<th>Intermediate</th>
<th>Complex</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>NLP-SIR</strong></td>
<td>136.13</td>
<td>118.10</td>
<td>97.19</td>
</tr>
<tr>
<td><strong>EXCEL</strong></td>
<td>147.45</td>
<td>134.05</td>
<td>157.28</td>
</tr>
<tr>
<td><strong>Difference</strong></td>
<td>11.32</td>
<td>15.94</td>
<td>60.09</td>
</tr>
</tbody>
</table>

Table 3: Average time per task (in seconds)

Table 3 shows the average amount of time that was spend by participants performing each task. It can be seen that for all categories of tasks, the participants who were using NLP-SIR, completed the tasks faster than those using Microsoft© Excel. As the complexity of the task increased, the savings made by participants using the NLP-SIR system also increased with the difference in time on complex tasks being the only statistically significant result.

It can be seen that participants using NLP-SIR spent most time on the easy tasks and less on the subsequent tasks. One reason for this is that participants had to write more information for the easier tasks. These tasks required users to write down the names of the golf courses that satisfied the given set of criteria. It is also believed that as these tasks were the first to be attempted, users spent some time learning to use the system during these tasks. Further analysis will be conducted in the future to determine the total time that was spent by participants interacting with the system.

**Average Number of Utterances per task**

During the experiment, participants did not always get the answer they wanted from the system on the first utterance. If this happened the participant would need to rephrase the utterance and try again. For this reason the average number of utterances required to complete each task has been evaluated.

It was found that during the experiment participants using the NLP-SIR system needed, on average, 1.97 utterances to complete each task. The highest number of utterances given to the system by any one participant on a single task was 12. The reason for this was that the participant looked for additional information beyond what was required for the task. Ultimately the correct answer was recorded by the participant.
On a number of occasions it was found that participants asked the same question, in the same way multiple times. This may have been due to the way the results are displayed in an alert box. When the alert box is shown on screen the user is not able to go back to the spreadsheet without first closing it. Therefore if a participant wanted to check the answer against the data before writing it down, they would need to close the alert box, look at the data and determine if the answer was correct and then ask the question again so that it could be displayed on screen while they wrote it down.

Types of utterances received

During the experiment participants used many different ways to ask for the same information. It was noticed that a number of participants used “Google” style queries where they only input the least amount of words necessary to express what they wanted, for example “Boone flat Golf Course” or “number of executive courses in each county”. Other participants wrote full statements, as if they would give to another person such as “How many flat golf courses are in boone?” or “How many executive courses are available in Marion?”. (The previous utterance is presented as it was input by the user to the system, including the misspelling of the word “available”).

In some instances the system was not able to interpret the utterances the participants provided. In such cases the system would not perform any actions and the participant would be required to rephrase the utterance. One example of this occurred when a participant was trying to find the most common type of terrain in the spreadsheet. The first utterance given by the user was “Most used terrain”. This utterance did not produce any results so the user then re-phrased the utterance to “Most popular terrain” to which the system responded with the correct answer.

It was also seen that a small number of participants copied the request directly into the NLP-SIR input box. On some occasions this approach produced the correct answer while on others the results produced were incorrect.

Limitations of the study

The results presented here are based on a well structured spreadsheet, which conforms to the assumptions that are made by the NLP-SIR system. Despite these limitations the results demonstrate that natural language is an effective means of retrieving information from spreadsheets. Further work on the NLP-SIR system may provide a more general NLI to spreadsheet information retrieval.

It is believed that the participants that took part in the study are representative of the majority of novice spreadsheet users. These participants had only a minimal amount of training in the features that were required for the tasks like the majority of novice spreadsheet users.

5. FUTURE WORK

During the experiment a number of suggestions were made as to how the technology could be improved. The present implementation of the system presents users with a separate window in which they can type their questions. During the task however, participants found that the constant switching between the Microsoft© Excel spreadsheet and the NLP-SIR window was annoying. In order to address this, the system could use a task pane similar to the one that was used to present the tasks, thus allowing users to
freely and easily move between the Microsoft© Excel spreadsheet and the NLP-SIR interface.

Although the participants found the system very helpful, they were not satisfied with the way the information was presented. The current system presents answers in an alert box, which needs to be closed before the user can go back to the spreadsheet. It was suggested that results, such as tables, should be placed on a blank area of the spreadsheet instead of the alert box. This approach will allow the user to interrogate the spreadsheet while looking at the results.

In its current implementation NLP-SIR only presents the final value of a count operation to the user. This approach does not allow the user to verify that the utterance has been understood correctly or to see which values have been included in the count. To address this issue an additional feature will be implemented to allow a user to click on the result of such a count operation and have the rows that have contributed to this value to be highlighted. This approach will allow the user to verify that the right criteria have been used in performing the count operation and thus increase users’ confidence in the answers produced by the system.

It is believed that some of the limiting assumptions that are made by NLP-SIR could be removed through further analysis of data spreadsheets.

The research to date has focused on novice spreadsheet users. Although these users will benefit most from the technology, they are not the only ones who will benefit. It is our belief that as the complexity of the task increases the benefits of a natural language approach will be seen by more experienced users. In order to validate this claim a similar experiment will be run in the future with expert spreadsheet users.

6. CONCLUSIONS

This paper introduces NLP-SIR, a Natural language approach to spreadsheet information retrieval. Natural Language Interfaces, such as NLP-SIR, allow users to operate computer applications through their own language, eliminating the need for users to conform to existing interfaces. Such approaches are more intuitive to users and encounter less resistance from new users.

An evaluation of NLP-SIR was conducted to investigate its effectiveness compared with Microsoft© Excel for information retrieval. This evaluation asked 41 Novice spreadsheet users to retrieve ten pieces of information from a given spreadsheet. Approximately half of the participants used Microsoft© Excel for the task while the remainder used the NLP-SIR system.

It was found that those participants who used the NLP-SIR system were able to successfully complete more of the tasks than those who used the existing application interface. It was also found that the NLP-SIR users were able to complete these tasks in less time.

REFERENCES


The Medical Algorithms Project

M Sriram Iyengar, PhD, Asst. Professor, School of Health Information Systems, UTHSCH. Email: m.sriram.iyengar@uth.tmc.edu, 281 793 4733, John R Svirbely, MD, TriHealth, Cincinnati, Ohio

Abstract:
The Medical Algorithms Project, a web-based resource located at www.medal.org, is the world’s largest collection of medical-related spreadsheets, consisting of over 13,500 Excel spreadsheets each encoding a medical algorithm from 45 different areas of medical practice. This free resource is in use worldwide with over 106,000 registered users as of March 1, 2009.

Introduction
A medical algorithm is any formula, score, scale, diagram, or computational technique that is useful in healthcare. Examples of medical algorithms include the Glasgow coma scale, TNM stage in oncology, and predictive risk factors for heart disease. Medical algorithms comprise a technology for medical decision support, with the potential to decrease time demands on clinicians, to support evidence-based medicine, to reduce errors, and to help increase the quality while decreasing the cost of care. We estimate that the biomedical peer-reviewed literature contains over 250,000 medical algorithms across all the specialties and sub-specialties of medicine. However, these algorithms are underutilized by busy clinicians because it takes time to find, to understand and to adapt them to a specific clinical situation. Our hope has been that they would be used more often by clinicians and medical researchers if they were readily accessible in a digital format.

The medical algorithms project (www.medal.org) was developed to enhance the availability and accessibility of medical algorithms to medical practitioners such as nurses, physicians, pharmacists, biomedical researchers and others in the healthcare field. This resource contains descriptions and executable versions for over 13,500 medical algorithms in 45 different subject areas (Fig. 1). All of the algorithms are implemented as Microsoft Excel spreadsheets; in addition, approximately 400 spreadsheets are implemented in PHP.

In the following we describe this resource further.
The medical algorithms project was conceived and developed by John Svirbely, MD, Trihealth Hospitals, Cincinnati, and Sriram Iyengar, PhD, School of Health Information Sciences (SHIS), University of Texas, Houston. They observed that large numbers of computational techniques such as scales and scores were being presented in the biomedical journals and books. These medical algorithms can potentially aid the practice of healthcare since they are derived from experimental or observational data. The numerous benefits for medical practice include support for medical decision-making, increased accuracy of diagnosis and prognosis, use of evidence-based criteria rather than subjective judgments, increased scope of physician practice and a reduction in medical errors. However, many medical algorithms are not used because few practitioners have the time to locate useful medical algorithms, let alone convert these from paper to computational formats. For these reasons the authors decided to create a searchable knowledge-base containing computable versions of paper-based medical algorithms. Clearly, there are many benefits to computer-based algorithms over manual execution using a calculator. These include error-free execution every time (provided the algorithm is programmed correctly!) especially for more complex formulae that include mathematical functions like exponentiation.
Implementation into Spreadsheets

This decision led to a search for the most suitable computing technology. After some discussion Excel spreadsheets were selected. There are several advantages to using spreadsheets:

1. Due to the dominance of Microsoft Office, spreadsheets are ubiquitous and available even on many mobile devices
2. The spreadsheet interface is familiar to many clinicians
3. The rich set of formulas built in to Excel spreadsheets means programming is not needed to encode medical algorithms
4. Spreadsheets are relatively simple to create, modify and maintain.
5. Helps and hints for using the algorithm can be easily added into the spreadsheet itself. However, detailed algorithm documentation and references require a separate word document.

The first online version of medal.org was released in 1998 with about 860 spreadsheets. These spreadsheets were divided across 14 chapters corresponding to topics in medicine such as neurology, oncology, gastroenterology and others. From the home page users were presented with a Table of Contents with one hyperlink entry for each chapter. Clicking on such an entry led the user to a page containing a list of algorithms for that chapter. Spreadsheets corresponding to a chapter were collected into a workbook enabling users to download an entire chapter at a time. However, in 2001 workbooks became impractical and unwieldy due to the rapid increase in the number of algorithms in each chapter as well as the increase in the number of chapters. For this reason the workbook format was replaced by one in which clicking on the name of an algorithm in a chapter enabled the user to either download the corresponding spreadsheet or execute it on line. In 2008, a custom program was used to convert the spreadsheets into ASPX pages; this preserved the spreadsheet metaphor and the ability to execute the spreadsheet as an ASPX (Active Server pages) server-side application.

An interesting use of Excel was that from 1998 to 2002 the entire HTML code for the medal.org web site was created by writing an HTML code generator in VBA (Visual Basic for Applications) for Excel. To achieve this a master spreadsheet was created that contained the names of all the algorithms and their chapter names, numbers and the file names of the sheets. This master sheet served as an index into a database of spreadsheets. The code generator was written in VBA and included as a macro in this master spreadsheet. For each version (a new version was created about once every 6 months) after populating the master spreadsheet with the information about new algorithms and spreadsheets the HTML code for the entire web site could be created and written out to disk by running this macro.
**Current Status**

The Medical Algorithms Project has been available for 11 years. Access is free, but as of September, 2004, new users must register for access. The material is directly linked (non-login access) from computers in the United States Veterans Administration (VA) healthcare system.

The site has about 1,600 unique visits every day and is the #1 result returned from a web search for the terms ‘medical algorithms’ or ‘algorithmic medicine’.

**Profiles of registered users**

As of March 2009, the site has about 106,900 registered users from over 200 countries. The majority of registered users are in the USA and the UK. Apart from papers about this resource at various venues such as AMIA (American Medical Informatics Association), no marketing has ever been done. Word of mouth and web search for specific topics are the typical ways in which the web site is found.

![Figure 2: Distribution of registered users by profession](image)

**Challenges**

As can be imagined in an ever-growing web site that currently includes 13,500 sheets, a number of challenges have been encountered over the years. Most of these have related to the specific algorithms such as the occasional correction to a formula or the issuance of a new version by the original creators. The spreadsheet format itself has provided to be robust and enduring; however, it has limitations for providing an enhanced user interface. The 400 algorithms written in PHP do provide these features.
A key concern has always been quality assurance with respect to correctness of formulas, use of outdated algorithm versions, keeping up with current literature and so on. Apart from these, so to speak, editorial issues, the resource has given rise to a nascent are of clinical informatics called *medical computation problems* that is concerned with optimal algorithms for finding the best algorithm(s) to apply in a given clinical context. Some issues in this regard are to match up algorithms and patients. For example, an algorithm designed for a pediatric population may give misleading information when used for an adult.

**Issues for Future Development**

The Medical Algorithm Projects continues to evolve and expand.

Future development will entail:
(1) a bidirectional interface to the medical record. This would overcome a serious limitation in that the current web site requires manual data entry. Manual data entry can be tedious and a source of preventable error.
(2) tools to improve the user experience, including automated selection
(3) integration into medical guidelines
(4) tools for data mining which can be used to improve processes or provide educational opportunities

It should be noted that, due to liability concerns, the spreadsheets are specifically approved only for teaching and research, not for patient care. Approval and validation to meet the good manufacturing requirements for a medical device as specified by the US FDA (Food and Drug Administration) is required to apply this decision support technology into actual patient care.

A recent development has been interest from publishers like Springer in producing specialized collections of medical algorithms in printed form. Springer has started issuing the Medal series on Medical Algorithms. The first one in the series is on Rheumatology and this has been issued in English, German, and Spanish.

**Conclusions**

Medical algorithms are a valuable but underutilized resource in healthcare. Their use by clinicians and medical researchers, especially at the point of care, can significantly improve the quality and cost-effectiveness of medical care. The Medical Algorithms Project encodes more than 13,500 medical algorithms for 45 areas of medicine into Excel spreadsheets and makes these available freely at the web site www.medal.org.

**References:**


13. Iyengar MS, Svirbely JR. The medical algorithms project: an online clinical information resource supporting medical decision-making. 128th Meeting of the Association of Clinical Scientists. May 14-17, 2008. Los Angeles, California.


From error detection to behaviour observation: first results from screen capture analysis

Françoise Tort, François-Marie Blondel, Éric Bruillard
UMR STEF – INRP – ENS Cachan, UniverSud
61, avenue du Président Wilson
94235 Cachan cedex - France
francoise.tort@ecogest.ens-cachan.fr
francois-marie.blondel@inrp.fr
eric.bruillard@creteil.iufm.fr

ABSTRACT

This paper deals with errors in using spreadsheets and analysis of automatic recording of user interaction with spreadsheets. After a review of literature devoted to spreadsheet errors, we advocate the importance of going from error detection to interaction behaviour analysis. We explain how we analyze screen captures and give the main results we have obtained using this specific methodology with secondary school students (N=24). Transcription provides general characteristics: time, sequence of performed tasks, unsuccessful attempts and user preferences. Analysis reveals preferred modes of actions (toolbar buttons or menu commands), ways of writing formulas, and typical approaches in searching for solutions. Time, rhythm and density appear to be promising indicators. We think such an approach (to analyze screen captures) could be used with more advanced spreadsheet users.

1. INTRODUCTION

Several studies have been devoted to the detection of errors and mistakes in spreadsheets, leading to some agreement concerning typology of errors (Panko & Halverson 1996; Chadwick & Sue 2001, Panko 2008). These studies have great importance for formulating strategies for identifying possible errors in spreadsheets. Are they completely conclusive?

One can imagine that errors are classical and stable in the sense that users in the eighties and “new millennium” users are expected to commit the same ones. But, these “new users” may have new ways of doing things, have acquired new routines leading to different kinds or patterns of errors. Adopting an educational point of view also leads to go beyond errors, notably to discover their origin.

During the DidaTab project (Blondel et al. 2008), we designed paper and computer tests to assess the competency levels of students. Observations made during these tests revealed that the way students work with spreadsheet software is as interesting as the final result they obtain. Their way of doing so is not always simple and is often surprising. Starting from these observations, we try to consider errors mainly as symptoms, inviting us to better understand the process underlying mistakes.

Furthermore it is useful to distinguish competency and performance. Not to be able to solve a particular question often is a weak indicator of understanding. The question is not only what students’ answers are but how they proceed, leading to other issues. Can we infer from the way they respond to tests what their level of understanding of spreadsheet
objects is? Do tested students show characteristic approaches and behaviours allowing us to better understand their way of working with software such as spreadsheets? What difficulties do they encounter?

Little empirical data are available to researchers to analyze the processes and actions of end-users.

In the first part of this paper, we give an overview of classic literature on spreadsheet errors and advocate the necessity to go more in-depth in analyzing the solving processes. In the second part, we describe different methodological approaches to get an outline of the user interaction process using spreadsheets. In the third part, we present the first results obtained in an experiment of screen capture analysis of students working with spreadsheets. Finally we propose some discussion.

2. SPREADSHEET ERRORS AND LEARNING PROCESSES

Panko and Halverson (1996) revisited by Panko (2008) offer a very complete taxonomy of spreadsheet errors. They distinguish between quantitative and qualitative errors. Quantitative errors lead to incorrect final values. Qualitative errors refer to risky practices that could lead to errors in later uses of the spreadsheet. The authors further divided the quantitative category, into mechanical errors, logic errors and omission errors. The classification criteria are relatively simple, and provide a reliable classification.

Rajalingham et al. (2000) propose a more complex taxonomy of spreadsheet errors. They make the same quantitative and qualitative distinction as Panko and Halverson. They distinguish accidental errors from reasoning errors (similar to mechanical errors and logic errors) and introduce an interesting difference between developer errors made during the development of the spreadsheet and end-user errors, covering errors made by users manipulating a spreadsheet after its creation.

Taxonomies allow the comparison of errors across studies and application areas. They are important to formulate strategies for discovering possible errors in operational spreadsheets, in order to fix them or prescribe a kind of cure.

In the educational context of students learning spreadsheets, we study abilities and difficulties of, and obstacles to, spreadsheet learners. Errors are mainly considered as symptoms and indicators of particular difficulties. Taxonomies can be used to diagnose the cause of errors. But the main point is not to detect errors and fix them, but to know how errors are created and what their causes are. Therefore, we are interested not only in students’ spreadsheet errors but also in the process of their creation.

2.1. Students’ spreadsheet errors

We will focus on two of the studies which have analysed students’ spreadsheet abilities, dealing with the possible origins of errors.

In an experimental study comparing the traditional spreadsheet calculation paradigm with a structured spreadsheet calculation paradigm, Tukiainen (2000) tried to explain errors made by novice users in terms of the computation “doctrine” (spreadsheet-based programming as opposed to other programming techniques), and of the spreadsheet calculation tool’s usability issues.

Reinhardt and Pillay (2005) aimed to develop a spreadsheet tutor helping students to overcome their learning difficulties. They studied what students were doing during a spreadsheet test in order to build an error library and a model to provide individualized
feedback. Their own classification focuses on errors in formulae and functions, and
distinguishes between three errors categories, depending on their level of generality:
errors that apply to any function and formula, errors that occur in formulae and functions
with a similar syntax, errors that only occur in specific formulae or functions. They
provide interesting and detailed examples.

In the framework of the DidaTab project, we administered computer-based tests to about
200 French secondary school students (Blondel et al., 2008). Students had to perform
very basic tasks: change cells format, write a formula (example in figure 1), create a
graphic or sort data (see Tort and Blondel 2007 for methodological details).

<p>| | | | | | | | | | | | | | | | | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>3</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></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>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>5</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>7</td>
<td>6</td>
<td>12</td>
<td>3</td>
<td>8</td>
<td>6</td>
<td>15</td>
<td>6</td>
<td>18</td>
<td>9</td>
<td>10</td>
<td>9</td>
<td>16</td>
<td>10</td>
<td>4</td>
<td>12</td>
<td>5</td>
</tr>
<tr>
<td>9</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>10</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

**Figure 1:** Task 3, Write the sum and the average of a vector

Our results revealed spreadsheet errors already detailed in (Tukiainen, 2000) or
(Reinhardt & Pillay, 2005) studies. Some errors we found are not described in the
literature and appear more difficult to classify in known taxonomies. Most of them were
made by initial spreadsheet users (Table 1).

1. Plotting a series of data available on a line, including the total amount which is
displayed at the end of the line.
2. Sorting only one column of data instead of the whole table of data.
3. Erasing digits after the decimal point of a number in a cell, in order to display
only one digit after the decimal point, instead of changing the display format of
the cell.
4. Using functions in non-conventional manners like SUM(A1+A2+A3) or
PRODUCT(A1*12,5) or SUM(A1*12,5).
5. Writing successive formulae with relative cell references, instead of writing one
formula with an absolute cell reference and copying it with the fill handle.

**Table 1:** Some errors of initial spreadsheet users observed during DidaTab project

Some errors can be considered as general errors. For instance, example 1 is the
transposition in the context of the creation of a graphic of the error GE4c "Additional
cells included in argument. Student included the title or additional cells at the end or at
the beginning of the necessary block of cells." described in (Reinhardt & Pillay, 2005)

According to the Panko and Halverson (1997) classification, examples 1 and 2 are
quantitative errors. They could be logical errors, if we assume that students applied a
wrong method. They could be mechanical errors, if students obtained such bad results
because they agreed to a suggestion automatically proposed by the spreadsheet software.
Example 3 is a pure quantitative logical error. Examples 4 and 5 are qualitative errors, not
visible from displayed values, but risky for later use of the spreadsheet.
All these errors are of high interest, but their origin is difficult to infer: traces of confusion of students while using the spreadsheet software, symptoms of students misunderstanding of spreadsheet objects and principles, or both.

Our aim is to investigate how these errors have been created and why.

2.2. Errors creation process of initial spreadsheet users

Powell et al. (2005) point out the fact that relatively little is known about the creation of spreadsheet errors. They notice that only a few laboratory experiments have been carried out, and no one has attempted to study spreadsheet development in the field, at a level of detail allowing observation of developers making errors.

Most of the students we have tested were relative novices in the use of spreadsheets, but had already personal experience of ICT. We can assume that these “new learners” have developed their own ways of working with new digital objects and computer applications. Chadwick and Sue (2001) observe that novices overrate their own literacy skills and make errors despite being quite confident. It is likely that they are satisfied most of the time by their own ways of doing things. We presume that their prior experience with other software is involved in the process of spreadsheet errors creation.

In the following section, we propose a methodological framework to observe learners creating spreadsheets at a very high level of detail.

3. METHODOLOGICAL APPROACHES

3.1. Data collection for process analysis

Creating spreadsheets may be considered as a particular case of software development. Human-Computer Interaction researchers have analysed user behaviour during software development with the intention of understanding what cognitive skills are involved.

In a paper on the various analysis that can be used to study the way users interact with a computer applications, Hulshof (2004) discusses the most commonly used methods to track interaction: eye tracking, thinking aloud and log files. Eye tracking provides researchers with precise information about the type of information users are processing, but interpretation may be difficult when high level cognitive processes are studied. The think-aloud method is well suited for studying users working on complex tasks but requires conditions and means that are not often available, particularly in a field study. Moreover, think-aloud protocol and related techniques may disturb the experimental subjects. Recording user actions in log files is rather easy to perform, and as a consequence, very popular in usability research, although interpretation of user reasoning may be more difficult than with other methods.

Most of these techniques have been used by researchers who study spreadsheet users. In one of the first experiment on users creating spreadsheets with Lotus 1-2-3 (Brown & Gould 1987), participants were videotaped and their keystrokes recorded using specific software. Beyond the elementary actions performed by users, video recording allows researchers to identify more precisely time spent for pausing and planning a solution. Ethnographic techniques mostly relying on interviews of spreadsheet users have been adopted by Nardi and Miller (1990) to study how people cooperate in creating and modifying spreadsheets. We used these techniques in the DidaTab project to record a detailed view of the uses of spreadsheets at home and in school by students. The most recent works rely on data-logging techniques, based on add-ons to the spreadsheet
software that records most of the user actions in log files. Adler and Nash (2004) developed a server based software tool (TellTable) to control and record control access, management and use of spreadsheets, and applied this tool for assessing students. In order to study professionals and students detecting and correcting errors, Brain and McDaid (2008) developed a VBA plug-in for Excel that records all cell selection and change actions. The main advantage of this non-intrusive technique is that it allows the recording of the actions of a significant number of participants without modifying their normal working environment.

In a learning context, it is not so easy to define in advance all meaningful actions of users. Although some students may repeat already known errors and processes, some of them may interact in new ways, depending on their usual practice with other computer applications. To cope with this open situation, we adopted a more exploratory approach. Our main objective was to trace most of student gestures in order to be able to guess some of the intentions behind the actions. And we wanted also to be able to replay all the actions performed by the users. With these objectives in mind, we choose a technique to record all events and actions appearing on the screen, even those which are not recorded by ordinary data-logging techniques: video recording of the screen.

### 3.2. Screen capture of screen events

Among several solutions that can be adopted to capture all actions appearing on the screen during interaction, we choose Camtasia Studio© by TechSmith. It captures all onscreen activities: mouse move, selection of a cell or of a range of cells, menu dropdown, click on a command in a menu, window move. In addition, the images taken from a webcam showing the user's face can be included as a “vignette” in the capture. The resulting videos may be saved in several formats and the volume of the corresponding files is reduced by means of a specific video codec.

A free Camtasia player is used to replay all recorded screens. Videos show all events that are traces of students’ gestures. In particular, the mouse moves on the screen can give some indications of the information that users are processing, similar to eye tracking methods.

Combined with the recording of the spreadsheet files which provide data concerning students’ production, screen video captures provides useful data for analysing students’ processes.

### 3.3. Transcription of events

The most important part of the interaction analysis relies on the transcription phase. The idea is to aggregate very elementary actions into more significant events, more suited for the intended level of analysis.

The screen capture allows detecting all visible changes appearing on the screen, whether they are produced by the user using the keyboard or the mouse, or by the computer application or the operating system as a response to an action.

In the particular case of the transcription of interaction occurring during a test on spreadsheet skills, the whole set of visible events is rather limited and may be identified.

An excerpt of such a transcription is given in Table 2. First, all student actions are reported in detail, except for some of them that are combined into one. For instance, “click on A7” followed by “track to V7” is combined into “select range A7:V7”. The feedback given by the spreadsheet software is also recorded with the action. For each
action or event, a comment giving additional details or suggesting an interpretation is often added. If no action occurs during a significant period of time, 10 seconds for instance, this “event” is also recorded in the transcription. Then, if a sequence of actions seems to be performed to reach a particular sub-goal, the sequence is aggregated into one step. The duration of the step is calculated from the start time of the first action of each step.

From our own experience, the number of detailed actions may vary from 5 to 30 for transcribing one task, depending on the complexity of the task and the activity of the student.

Specific tools like Transana or ActivityLens (Avouris et al., 2007) may be used to assist researchers in that task.

<table>
<thead>
<tr>
<th>Start Time (h:min:sec)</th>
<th>Action / Event</th>
<th>Result of action</th>
<th>Comment</th>
<th>Duration (h:min:sec)</th>
<th>Step</th>
</tr>
</thead>
<tbody>
<tr>
<td>0:01:52</td>
<td>no mouse move</td>
<td></td>
<td>Reads question?</td>
<td>0:00:21</td>
<td>Reading</td>
</tr>
<tr>
<td>0:02:13</td>
<td>click on cell X9</td>
<td></td>
<td>Text cursor appears in the cell</td>
<td>0:00:34</td>
<td>Typing &quot;6+12+..&quot; stopped</td>
</tr>
<tr>
<td>0:02:17</td>
<td>keystrokes in X9: &quot;=6+12+3+...+13&quot;</td>
<td></td>
<td>Develops a solution then stops</td>
<td>0:00:16</td>
<td>Thinking</td>
</tr>
<tr>
<td>0:02:47</td>
<td>no mouse move</td>
<td>Text cursor blinks at the end of the formula</td>
<td>Thinks of a solution?</td>
<td>0:00:25</td>
<td>Inserting a formula stopped</td>
</tr>
<tr>
<td>0:03:03</td>
<td>delete until char =</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>0:03:22</td>
<td>click on the menu close to the sigma button</td>
<td>menu showing current functions drops down</td>
<td>Not enough time to read</td>
<td>0:00:25</td>
<td></td>
</tr>
<tr>
<td>0:03:23</td>
<td>..</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

**Table 2:** Transcript excerpt of the screen capture of student P8, task 3 (see Fig. 1)

3.4. General characteristics collected by transcription

Four main characteristics orient the transcription process. Examples are taken from task 3 (see table 2 and figure 1).

**Time**

Using the start time of actions, it is possible to calculate the time spent on every task, the duration of every step and also of inactive periods, and observe possible increases or decreases in the rate of actions during the whole test. For instance, student P8 has spent 7’ on task 3.

**Sequence of performed tasks**

The sequence of tasks which are successively studied can show navigation between tasks with forward or backward moves. For instance, student P8 left task 3 after several attempts lasting 3’ 21”, solved task 7 in 1’07” then returned to task 3 for 3’46”.

**Unsuccessful attempts**

All unsuccessful attempts that leave no traces in the files can be observed. For instance, when working on task 3, student P8 first typed a formula with literals then tried to use a predefined function, then tried to paste the A7:V7 range into cell X9 and finally wrote a formula adding the 22 cell references.
Software interface uses

User preferences in using software interface features can be easily observed. For instance, student P8 entered the formula in task 3 by typing the + sign and selecting cells with a mouse click; student P2 entered the same formula by typing all characters, including cell references.

4. EXPERIMENTS

4.1. Tests

Tests are built using the methodological framework of the DidaTab project (Tort & Blondel, 2007). Each test includes a series of tasks distributed among five categories of spreadsheet skills. In each task, students are given a spreadsheet that contains data taken from a very simple real life example. They have to either change or transform the appearance of the given spreadsheet (enlarge a column, bold some cells, sort columns), or make calculations from data, or create new objects like charts or tables.

Tests used in the experiment included 11 tasks, and lasted about one hour. Tasks were distributed over skills categories in the following way:

- Editing Cells and Sheets: two tasks (Task 2 Fig. 2 below);
- Writing formulas: five tasks, that require the student to write formulas with arithmetic operators, relative cells references, absolute cells references, functions (Task 3 Fig. 1), to recopy formulas with absolute cells references (Task 7 Fig. 3);
- Creating graphs: two tasks, one that requires creating a curve graph and the other choosing the suitable graphs for two ranges of values;
- Managing data tables: one task that requires sorting a data table on a single criteria (Task 8 Fig. 4 below);
- Modelling: one task that requires building a series of dates.

![Figure 2: Task 2, Change data display](image)

![Figure 3: Task 7, Calculate a tax using absolute reference.](image)

![Figure 4: Task 8, Simple sort.](image)

As Brown & Gould (1987) and Reinhart and Pillay (2005), we found in a previous study (Tort et al. 2008) that students experience the most difficulties when they have to write formulae. We wanted to know if the observation of the process by which they deal with
writing formulae would give us information on such difficulties. That is the reason why half of the tasks focussed on writing formulae.

4.2. Procedure and data collection
Tests have been administered to two groups of students from the same senior high school, situated in a small town in a suburb of Paris:
- 31 students in the first year of a “sciences” stream of study, aged 16-17 (grade 11, SC group).
- 12 students in the second year of a “social science and economics” stream of study, aged 17-18 (grade 12, SE group).

Tests were administered on computers, at school, during school time, by the mathematics teacher, who is the same for both groups. The teacher told us that students of the “social science and economics” (SE) group used spreadsheet software during his lessons the year before. Students in the “sciences” (SC) group did not use spreadsheet in his lessons, but about three quarters of them attended, the year before, an optional course on data processing in physics and chemistry, where they were introduced to spreadsheets and computer programming. It should be also noticed that because of their good level in mathematics, students in the SC group were expected to have few difficulties in using a spreadsheet (on the link between mathematics and spreadsheet abilities see Reinhart & Pillay (2005) and Haspekian & Bruillard (2007).

The application software was Microsoft Excel© 2003. The interface was absolutely the same for both groups, especially displayed toolbars.

Students had to work directly within a given spreadsheet file. An explanation of the whole procedure was given on the first sheet, and the tasks, numbered 1 to 11, were distributed in the next 11 sheets. Students saved their files at the end of the test. Screens were recorded with Camtasia Studio© TechSmith, during the whole duration of the test.

The 31 students in the sciences (SC) group and the 12 students in the social sciences and economics (SE) group completed the whole tests without any incident; 43 (31+12) spreadsheet files and 43 screen video records were collected. The duration of records extends from 25 minutes to 42 minutes with a mean of 38 minutes.

4.3. Abilities revealed by productions
In Tort & al. (2007), we reported on previous tests administered to four groups of students. We found that all the students succeeded in tasks dealing with superficial manipulations of spreadsheet objects that do not need specific knowledge, especially tasks linked to formatting cells. All results decrease slightly when the tasks require more understanding of spreadsheet objects and functionalities. The same is the case for tasks concerning graphs and tables. Moreover, these tasks are more discriminating among the different groups’ results. Lastly, the more discriminating tasks where those which dealt with writing formulas.

The results of the two groups studied in the present paper are significantly better (see Tab. 3). In particular, students in the SC group had better results on tasks dealing with objects specific to spreadsheets: decimal number format, cells copying, functions, choosing a suitable graphic and data sorting. Also, the students in the SE group had good results for entity manipulations, writing simple formulas, and graphics. Nevertheless, they had lower results on formulas with functions, absolute cells references, cells copying and data sorting.
5. FIRST RESULTS

We present in this section a first exploratory study of screen records analysis and give main results we have obtained using this specific methodology. We decided to study more in-depth the same number of students in each group for some comparisons and to keep 12 students of each group (total, N=24).

The next section will be devoted to a discussion about this methodology.

The analysis of screen video records was guided by two major aims.

Firstly, we systematically identified the modes of actions used by students on the software interface. It appears that each group of students had specific preferred modes of action: menu commands vs toolbar buttons, keyboard vs mouse, and the preferred actions depend on the performed tasks. It seems that students reproduced actions they have learned.

Secondly, we paid attention to students who did not know how to perform tasks. We look for similar behaviours observed for different students and different tasks.

5.1. Preferred modes of action

**Toolbar buttons or menu commands**

Spreadsheet software offers different ways to activate a command: menu commands, contextual menus, keyboard shortcuts and toolbar buttons. Toolbar buttons activate commands with default settings, depending on the current selection. Menus and keyboard shortcuts open dialog boxes, where users can choose the parameters of the command.

Screen videos show that the two tested groups used different modes of action to change cell format (Task 2) and to sort data (Task 8). Students in the SC group used the menu commands and students in the SE group used toolbar buttons (See Table 4).

<table>
<thead>
<tr>
<th>Task</th>
<th>Actions</th>
<th>SC group (12)</th>
<th>SE group (12)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Task 2</td>
<td>Decrease Decimal button</td>
<td>0</td>
<td>10</td>
</tr>
<tr>
<td></td>
<td>Format cell menu command</td>
<td>11</td>
<td>0</td>
</tr>
<tr>
<td>Task 8</td>
<td>Sort button</td>
<td>3</td>
<td>10</td>
</tr>
<tr>
<td></td>
<td>Data/Sort menu command</td>
<td>6</td>
<td>1</td>
</tr>
</tbody>
</table>

Table 4: Approaches used to activate commands - tasks 2 & 8
Moreover, students who easily solved these tasks worked very quickly: less than 50 seconds to change cell format and less than 2 minutes to sort data. We assume that they all repeat a familiar gesture previously learned.

Students who took time to perform the tasks used the toolbar buttons:
- For task 2, screen videos showed 3 students took from 2 to 6 minutes to find the Decrease Decimal button, which was not displayed in the toolbar. Then they were lost because the button changes its place once it has been used.
- For task 8, 11 students made several trials to sort data. All of them use toolbar buttons.

**Writing formula in the bar formula or in the cell**

In spreadsheet software, the Formula bar is used to enter and modify the formula of the selected cell. This bar contains several buttons to cancel, to save the input, to open the library of functions. Another way to put a formula in a cell is to type directly the formula in the cell. According to us, this second method maintains the confusion between the displayed value of the cell and its actual content.

<table>
<thead>
<tr>
<th>Task</th>
<th>Actions</th>
<th>SC group (12)</th>
<th>SE group (12)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Task 3</td>
<td>Entry in the Formula bar</td>
<td>8</td>
<td>4</td>
</tr>
<tr>
<td></td>
<td>Entry in the cell</td>
<td>4</td>
<td>8</td>
</tr>
</tbody>
</table>

**Table 5: Approaches used to enter formulas – Task 3**

Screen videos show that half students used the formula bar, most often in the SC group.

**Inserting a function in one click or navigating in the library of functions**

The ∑ button is used to insert a function in few mouse clicks. One click on the button inserts the expression "=SUM()" in the selected cell. Moreover, when the selected cell is at the end of a range, the expression is supplemented by the cell references of the range (for instance "=SUM(A7:V7)"). In task 3, such an approach does not work because the required formula is not situated at the end of the data to be summed.

Another method consists of using the "Insert a function" menu command. It opens a dialog box, which helps the user to browse the library of functions, choose a function and fill its arguments. A third method consists in directly typing the formula in the cell or in the formula bar. This last method is only useful for users who know the name and the syntax of the function.

<table>
<thead>
<tr>
<th>Task</th>
<th>Actions</th>
<th>SC group (12)</th>
<th>SE group (12)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Task 3</td>
<td>&quot;sigma&quot; button and mouse selection of cells to be referenced</td>
<td>3</td>
<td>1</td>
</tr>
<tr>
<td></td>
<td><em>Insert a function</em> menu command</td>
<td>3</td>
<td>0</td>
</tr>
<tr>
<td></td>
<td>keyboard input of the entire formula</td>
<td>3</td>
<td>1</td>
</tr>
<tr>
<td></td>
<td>&quot;sigma&quot; button and keyboard input of cell references</td>
<td>2</td>
<td>0</td>
</tr>
</tbody>
</table>

**Table 6: Approaches used to insert a SUM function – Task 3**

Screen videos showed that students who succeed in inserting a function use various approaches (See table 6). All these students made several trials before obtaining the final formula. Moreover, 4 of them (not reported in the table) failed in the use of the ∑ button.
Screen videos showed that most students have not gained a series of methods to perform usual spreadsheet tasks. We presume that they do not follow a well-known step by step process. More likely, their idea of functions is unclear and they need several trials with the interface to succeed in inserting a function.

Discussion

We noticed that students who quickly solved a task seemed to reproduce methods that were familiar to them. In addition, each group favoured a different mode of action. Students in the SE group seemed to have experienced direct manipulation and often searched for solutions using this mode of action. Students in the SC group seemed to be more familiar with menus and dialog boxes. This mode of action (menus and dialog boxes) that allows a more detailed control of action is easier to re-apply to various computer applications. According to what we know about the curriculum, it is likely that the SC group spent more time using computers at school; and that the preferred mode of action of these students is partly induced by the type of computer applications they used.

5.2. Typical approaches in searching for solutions

In this second part of the analysis, we try to identify the most frequent ways students took when they did not know how to perform a task. The first results of this exploratory work are given below.

Systematic menu browse

In this kind of approach, which we named systematic menu browse, all menus, options and buttons are browsed. It seems that students take time to read the caption of every option. And if they try one command without obtaining success, they immediately come back to the last seen menu item or button. It looks as if students were looking for one unique command that solved the task. We assume that these students reasoned that there is necessarily one unique command per task to perform, even if a sequence of actions is required.

Trials of select-command combinations

We observed students who selected several different cell ranges before applying a command, by way of a button or a menu item. For instance, in the sort task, some students seemed to know which command to apply but several times, they changed the selected cells or the sort button. These trials show a difficulty for students to choose the correct set of data depending on a particular action. This difficulty is reinforced by the spreadsheet software interface which in some cases suggests using another selection before executing the command.

Opportunistic discovery by analogy

In these cases, students acted using analogy between commands. When working on a task, they “discover” a command that they think applicable to solve a previously uncompleted task and then go backwards to the previous task and try the “new” command. We may assume that this kind of analogy is often suggested by the names given to the options of menus. For instance, the label function and the choice sum in the dialog box opened by the menu Data Consolidate may introduce confusion for students trying to sum a cell range in task 3.
Time, rhythm and density

The three various approaches described above (systematic menu browse, trials of select-command combinations, opportunistic discovery by analogy) have been observed for several students working on several tasks. All these attempts take time and seldom end with a solution.

Moreover, when looking at the screen captures corresponding to these trial and error steps, one can observe that the mouse is always moving and actions follow one another at a high rhythm. In fact, the density of actions, i.e. the number of action per time unit, is greater in these trial and error steps than in other steps.

Within this population of students, those who succeeded in finding a solution after several attempts presented some peculiar ways of working with tasks:

- They spent less time evaluating a result, probably knowing more precisely what they want to see;
- They spent more time reading the content of dialog boxes and messages boxes before choosing an answer.
- They modified their sheets to fix the errors without deviating from the general purpose of the task.
- They immediately repeated a procedure just found.

6. DISCUSSION ON BEHAVIOUR RESOLUTION AND FUTURE WORKS

The analysis of screen captures shows a large amount of detail on the various ways adopted by users when working with spreadsheets. Applied to students, it reveals already known defaults and difficulties but also some particular modes of interaction with spreadsheet software.

A rather common behaviour relies on a student belief that each problem has one solution and that this solution has already been scripted in the application software. Some students work as if they just have to find the right button or the right menu item to perform a task. And this could explain what we named “systematic menu browsing”. This behaviour can be linked to a common belief of Google users that you just have to ask Google and Google will give you the answer.

Another common behaviour may be seen as a generalisation of “copy and paste”. Some students do not base their approach on a step by step process or a model of what has to be done, but on the recall of a similar problem followed by an adaptation with trials and errors. Some evidence of this behaviour may be found in what we named “opportunistic discovery by analogy”.

We guess that other interesting ways may be found that are linked to actual modes of interaction with computer applications of the new generation of students. Observation of diverse populations of students would certainly be helpful but it requires important means and screen capture analysis is time consuming. Anyway, it should be useful to establish a catalogue of the kinds of actions carried out by users using spreadsheet software. This could be linked with their level of understanding of spreadsheet objects required to perform these actions and with their practice with other computer applications.

Exploration of duration, rhythm and density has also to be continued. It seems that when users do absolutely not know how to solve a question, they interact very little with the software; when they perfectly know how to solve a question, they perform very few and well organised actions. Between these two extreme cases, users interact a lot more with
the software. The density of actions (number of actions / time) seems to be an interesting indicator to investigate.

The results of the exploratory study presented above can be important for the design of a curriculum on spreadsheets. We think that the methods we developed could also be useful to get a better understanding of the various ways professional users are working. Knowing what kind of behaviour users generally adopt in their professional work, depending on specific situations, can be useful in designing appropriate initial or in-service training.

7. REFERENCES


This page intentionally blank
Milestones for Teaching the Spreadsheet Program

Étienne Vandeput
CRIFA – STE – FAPSE – ULg, boulevard du Rectorat 5, B – 5000 Liège (Belgium) evandeput@ulg.ac.be

ABSTRACT

There are different manners of teaching a spreadsheet program. In any case, it is intended that the teacher settles the objectives of the course and adapts them to the particular audience he/she has to deal with. This paper aims at providing any teacher whatever his/her specific objectives and his/her audience with elements to help him/her building a course. It focuses mainly on two important issues: 1 - select in all that may be said about such complex tools, what is prior to know and to teach, i.e. what leads to autonomy in using but also to autonomy in learning (because everything cannot be taught) and 2 - show how concepts are closely related to good formatting considerations. A method based on the “invariants of information processing” is outlined, partially illustrated and an implementation is described throughout a course designed for students preparing a master in Education Sciences.

1 INTRODUCTION

Talks around « must one teach something, and if so, what about office automation tools? » have been dealt with for years. In Belgian secondary schools, by the end of the eighties, many teachers decided to renounce teaching programming because it seemed too hard. Most of them found that teaching about office automation tools with ICT was more interesting for students and more in relationship with what they would be asked to do later. They were right and wrong at the same time. Of course, teaching the art of programming is not so easy. It is probably the main reason why they gave it up. But they got the feeling that teaching office automation tools was simple and limited to menus exploration as well as to exercises with an online help or a user's guide consultation. We never thought so.

On the other hand, a spreadsheet program is more complex than a word processing program or a presentation program because processing still remains more formal, even generic and allows applications in a very wide range of domains. So, we do not have any reason to say that “There is nothing to teach about using a spreadsheet program.”.

In its 2nd section, this paper proposes to define what general objectives of a course on the spreadsheet program should be and to point out different elements that cannot be ignored to reach such objectives. The definition of the milestones of a course on the spreadsheet program is made easier using a method based on the “invariants of information processing”. The method is explained and partially implemented. Further, in section 3, we illustrate an implementation of these steps in a course sequence dedicated to students preparing a master in education sciences. A majority of them will need to use a spreadsheet program in the framework of his/her master's dissertation. Section 4 suggests a conclusion giving a feedback on the method developed in section 2 and guidelines inspired by the examples shown in section 3.
2 THEORETICAL REFLECTIONS

2.1 Short Considerations About Learning

To begin with, we think that some learning processes should be compared. Nowadays, teachers may use a range of strategies and tools to build their course scenario. E-learning, blended learning and so on provide them with a variety of environments and contexts they can use.

Of course, there are many ways leading to learning. And the depth of learning may also vary. Using a tutorial does not mean training, and training is not really the same as teaching. However, ICT didactical and pedagogical considerations are linked to such ambiguities. Many people (and unfortunately also teachers and trainers) sometimes consider teaching, training and using a tutorial as identical things. In this section, we will quickly try to explain what differences can be identified between these three ways of “making learn”. In the following sections, we will insist on what steps can be considered as fundamental in case teaching is the chosen way to make learn.

Tutorials

ICT learning often consists in ICT e-learning. In fact, if ICT exist, why don't we use them to learn, and particularly to learn about them? Teachers and trainers have the opportunity of creating learning supports like CDs, web sites, web applications... including various multimedia items like pictures, videos, animations, screen casts... That is what we could call “the big temptation”. The process becomes attractive and many people think that motivation grows when the learner is sitting in front of a computer screen. They are probably right. But, what about learning?

Using such a system makes it easy to « show ». So, the teacher (designer) can center the learning on the impregnation/modeling paradigm but also provide the learner with some pieces of advice developing the practice/guidance paradigm [Denis & Leclercq, 1994]. Though it is not a general rule, a reflection process is often missing because a tutorial generally emphasizes demonstrative aspects. The trainer entrusts the system with his/her own role, losing part of the interaction richness.

Notice that: tutorials, screen casts and other similar tools are often more useful when they are included in a more global pedagogical scenario initiated by a teacher.

Training

Even when the learning is not from a distance, the use of software in order to carry out a task is often considered like a practical process devoid of any intelligent approach. This point of view is unfortunately shared by a majority of learners and teachers. For instance, the use of a word processing program is considered by lots of people like a sequence of elementary commands. Such an approach cannot lead to a structured and efficient work. Nevertheless, many trainers build their training on this methodology. What the learner can see remains very important. So the trainer will insist on the graphical elements of the environment (menus, buttons, checkboxes…) and on graphical aspects of the process results. Examples about the spreadsheet are numerous. For instance, trainees learn to enlarge a column or to emphasize characters before writing a formula or thinking about the modeling of a spreadsheet. They are very surprised when a value does not appear in a cell like they thought it would, but they do not care about cell format and information
types. They feel very satisfied when they get a pretty graphic even if it shows wrong values and so on.

Vocational training often has precise and operational objectives and aims at developing professional skills. Sometimes, the trainees have to learn special routines related to their current or future job or activity. So, reflecting about the deep nature of things is not always a prior concern. That is why many trainers often hesitate to build their training on long-term effects and insist more on practical results rather than on competences gathering knowledge and know-how.

Teaching

If training often refers to a learning activity parallel to or following studies (sometimes from afar), with very precise objectives and sometimes expressed in terms of technical know-how, teaching rather refers to studies leading to a diploma. The objectives for the learners are related to more global competences. That means a much more long-term view. Skills to be developed are identified, but the processes aren't especially determined. Of course, this paper is dedicated to teaching in an academic context. So, let us develop in the next sections our point of view on teaching.

2.2 Objectives of a course on the spreadsheet program

Regardless of any audience, one can agree on a very general objective: make and keep the learners autonomous in the use but also in the learning of a spreadsheet program.

What does that mean?

The first part is easier to understand than the second one. In fact, the learner should offer proof that he/she is able to summon up his/her knowledge of the spreadsheet program to solve all kinds of problems related to the management of spreadsheets: building, modifying, correcting... Such an objective can easily be detailed into more operational ones. It is not so complicated to identify which knowledge (a small part depending on the audience) should be mastered. Trails are provided in the next section.

The second objective is more difficult to reach. The spreadsheet program, like many other ones, is very complex. It is well known that many people use only a few features among those that are offered. Some of these features are sometimes so specific that users need them in very particular cases. So, for a user (learner), it’s less important to know a lot about a software package than to be able to discover a feature when needed, because he/she has got good representations of how the global task can been formalized and how a system works. In other words, providing learners with such good representations (if needed) is quite essential. We find particularly important to insist on the formal size of numerical processing. It helps the learner to understand and sometimes to imagine how the system works or could work. We have developed this point of view in [Vandeput, 2006]. Some examples are given in the experience developed at the end of the present paper. These clearly illustrate that fundamental constraint due to any computer-based systems.

Let us come back to the first objective. There are two sub-objectives to reach and they are complementary. The first one is to master the basic concepts. Notice that basic does not mean simple! The second one is to be able to format a spreadsheet efficiently. Of course, this efficiency concerns the flexibility of the template (e.g. What if that data is modified?
Does the spreadsheet fit with changes?), but also the quality of information presentation (special data, results, graphics...).

The first is prior to the second. A learner will be unable to format a spreadsheet if he/she does not master the main concepts. As you will see in the final illustration, it does not mean formatting should be moved to the end of the course. Formatting is a know-how that can be developed at any stage of the learning.

To identify the basic concepts, we use a method which is applicable to any software package formalizing a task. It is based on what we call “invariants of information processing”. This method is applied to the spreadsheet program in the following section.

**2.3 Method to elicit basic knowledge**

**Invariants of information processing**

The method and its advantages are developed in [Poisseroux & al., 2008]. We shortly provide some explanations about it.

Obviously, the term “invariant” is used to call “what does not vary, what stays constant, fixed or stable”. In the domain of information and communication technologies (ICT), identifying invariants is useful. Users can focus on their goals (how to reach them efficiently) and not on what they can see (graphical environment). This way of thinking makes them become independent of the software interface and further, more autonomous using, but also learning about software.

Invariants can be either global, or related to specific software (e.g. modeling a task such as mail, document edition, information retrieval, management...). They may have two different natures. They are either concepts or (tasks) organizing principles. An organizing principle describes an automatic or semi-automatic process carried out by the system (any association “computer-software” at a particular moment of a work session) and dedicated to a task (e.g. word processing) or a set of tasks (e.g. office automation). An organizing principle is a specific software feature intended to shorten the user's task. Here are some examples.

Automatic calculation is a basic and very important organizing principle of the spreadsheet program. Power typing in word processing is another good example. Many programs implement automatic data capture (browser, spreadsheet, email...). The user does not need to encode long character strings (URL, data in a cell, email address...).

Organizing principles are invariants by nature because they are closely related to the task that does not fundamentally -or so slowly- evolve. Here are some examples. It sounds like an evidence that a spreadsheet implements automatic calculation or that word processing implements word wrap. Principles are stated using concepts. Because of this relation, concepts must be qualified as invariants. A paragraph (document edition), a slide (presentation), a cell, a formula (spreadsheet)... are concepts adopted by most of the software developers.

From a pedagogical point of view, invariants are pieces of knowledge. Knowing them should have a good influence on the user know-how, helping him/her to manage efficiently the system.

Like for other pieces of software, the spreadsheet developer has to formalize a task. Quickly and simply said, he/she has to imagine what kinds of goals the user will formulate when carrying out this task.
Spreadsheet program invariants

As a strategy, we could decide to identify a large number of invariants and to classify them into different categories. This has been done in [Vandeput & Colinet, 2005]. But there is another way to use invariants, that is closer to a didactical approach. That consists in:

1. starting from the more important organizing principles (the more related to the task, the more useful from an information processing point of view, the software core activity);
2. deriving the main concepts linked to these principles;
3. identifying other principles and concepts inspired by these concepts;
4. iterating steps 2 and 3.

This way of thinking helps defining a framework to build a course. Indeed, one important question when giving a course whatever it is, and this issue is particularly hard in the ICT domains (where things to be known have not been filtered and classified), is to determine which concept should be introduced first, and what about the next one, and the following one... Here is how we could proceed with the spreadsheet program.

As a matter of fact, the main principle organizing work with the spreadsheet is the recalculation principle, responsible for a spread of changes. It could be formulated as follows:

“The system (computer + spreadsheet program + OS of course) is able to automatically recalculate any value (piece of information) contained in a cell and depending, through a formula, on other values (pieces of information) contained in other cells of the same sheet, the same spreadsheet or possibly other spreadsheets”.

This principle should be defined more precisely but it contains all we need to explore the first invariant concepts related to it. Notice that each principle can be formulated in a sentence starting by “The system is able to automatically...“

Some concepts are entities: cell, sheet, spreadsheet... Most of software packages are based on entities handlings: word, paragraph, page... (word processing program), slide, title,... (presentation program), cell, column, row, sheet,... (spreadsheet program). Early identifying those entities is necessary because they are important pieces of the puzzle. The learner will have to know what kind of features may be applied to them and what are their attributes. Of course, entities are invariant concepts. Every spreadsheet program uses them in the same way. Some short differences may be observed as far as the features and/or the software instructions are concerned but not on the entities and how they are named.

Another important concept mentioned in the statement of the principle is the formula. When thinking about what is a formula, new concepts appear. Reflecting upon what makes up a formula, the response is very simple. A formula is an expression including some of the following elements: operators, constant values, (possibly) variable values and/or functions.

Values and variable make one think about types but variable also introduces the reference principle.

Notice that only four kinds of elements are used to build a formula. So, it is not
complicated to understand the components of the formula and to identify them easily. Identically, there are lots of functions but the learner does not need to know all of them. He/she must know the general syntax of a function. So, new concepts appear such as, identifier (of the function), arguments, separators...

The type concept evokes the very transversal coding principle that can be defined like this: “The system is able to code (digitize) information depending on what kind of operations will be applied on it and to decode digitized information so that it can be displayed.”. Notice that few types exist and that they can be grasped at the same time.

The type concept is inseparable from the formatting principle. One can say: “The system is able to display information in a default format or any other format defined and/or indicated by the user.”

At this stage we reach a kind of “Pons Asinorum”. Indeed, if the user (learner) encodes data before any formatting step, the spreadsheet program chooses by itself the type, the display format and the layout format, which can sometimes be very amazing. So, all these concepts must be clearly understood.

Once the reference principle is introduced, another principle arises: the (re)copy principle. This principle is connected to the following concepts: absolute reference, relative reference and reference by name.

That quick brainstorming shows how principles may evoke concepts and concepts may evoke other principles. A course sequence can be based on such a brainstorming. We recommend to provide illustrations and to make exercises introducing these concepts and principles one by one.

**Good Practice Guidelines**

The building of a « well-formatted» spreadsheet is related to a good knowledge of the invariants. For instance, the (re)copy principle has an influence on the spreadsheet design.

Thinking about the format principle, the good designer will determine him/herself the type and display format for all the significant parts of the sheets.

Another principle like the conditional formatting principle is also a tool to exploit in order to put some data in evidence and increase the quality of data presentation.

In the following sections, we develop and illustrate the didactical approach mixing the discovery of spreadsheet program invariants and the drawing up of guidelines.

### 3 CASE OF STUDENTS IN EDUCATION SCIENCES

#### 3.1 Context Description

At the University of Liege (ULg - Belgium), teachers, social and youth workers are given the opportunity of applying for a master in Educational Sciences. But, in order to get the access to the master without coming from a bachelor degree in the domain, they first have to follow a « one-year preliminary program leading to the Master in Educational Sciences ». This foundation course is composed of different courses aiming at providing them with some necessary basis. One of these courses is called « Introduction to the computerized processing of quantitative and qualitative data ». It should, in the future,
help students analyzing their data with, for instance, making reports or their master's dissertation. As it is a flexible and performing tool, an important part of this course is dedicated to the spreadsheet program.

Let us characterize the audience more precisely. Students are preparing a Master in Educational Sciences. Next year, they will have to choose a master's dissertation topic. This dissertation is, most of the time, an exploratory survey needing data analysis, graphical representations and related comments. That is why they often need to use a spreadsheet program. Most of them have a very short experience in the use of this kind of program. On top of that, some of them do not have any idea of what a spreadsheet program can be. The course aims at providing them, quickly, with good bases to lead such a work efficiently.

3.2 Didactical Approach

« What should be taught, and how? » is a relevant and certainly not a trivial question.

The objective of the course is very clear. Students should demonstrate their efficient use of a spreadsheet program in order to analyze, graphically present, and implement formulas to data. The duration of the course is 20 hours including theory and practical work. Time is short. So, we have to think about a didactical trail adapted to the situation.

This section describes how activities like teaching concepts (those that have been quickly evoked in the previous section), providing guidelines, carrying out formative evaluation are mixed in order to achieve the objective mentioned.

In order to make the learning process shorter and more efficient, we focus on two concerns:

- the course content (what students must absolutely know about a spreadsheet program);
- the art of building a spreadsheet.

Both concerns may (must) be dealt with simultaneously. The second point, of course, focuses on the main recommendations in relationship with the tasks that the students are likely to realize in the future. The teacher will find a very complete inspiration source in [Read & Batson, 1999].

Here are, accompanied with some comments, ten steps of the didactical trail. The pedagogical scenario is very simple. It includes, for each trail, an alternation of theoretical discussions and practical sessions.

First Step: Computer-Based Systems are Processing Information Formally.

Because of the low knowledge level of the students (a quick survey provided this information), the step is unavoidable. A session is devoted to making them understand well how a numerical system works. At the end, students must have understood that « in order to be processed by a computer, information must be digitized ». They should become familiar with coding and understand that coding has an influence on processings that, as a consequence, always concern information format and never its meaning.

Various situations are described. Here are some examples:

Same meaning but different formats:
Six, SIX, six, VI...
Computer, COMPUTER, ComPutER, computter...

Same format but different meanings:
...a train to Paris..., ...my train of thought..., ...to train or to teach...

Sometimes, frequent problems are used to underline the importance of coding like in the following example (extract from a French email message including stressed characters)

Bonjour Avant tout une bonne ann=E9e 2007 =E0 toutes et =E0 tous. Ceci =E9tant dit, passons aux choses p=E9nibles. Les accords de d=E9cembre pass=E9s par Marie Arena limitent les heures “non enseign=E9es”.

Second Step: Formatting of a Spreadsheet from an Existing Document Starts with an “Atomization” of Information.

Just an example to show that converting an existing document into a table is not so trivial.

The framework rows-columns does not appear. But another issue is to know how information must be broken down like for instance: « A payer avant le 06 novembre 2001 ». One cell, two cells ore more?

Here is a first opportunity to speak about good formatting.

Third Step: the User (Learner) Should Have the Control, not the System.

Exercises are proposed to show the learners that the system must be controlled. They are invited to put some particular character strings in the cells of a sheet and have to understand why the system displays something different from what they were thinking about.
Here are such strings. You can introduce them, one by one, in the cells of a sheet. What is really displayed is often amazing. What happens? The explanation is not always very simple.

123 12/3 12,3,4567 12-3 12+3 12/12/12 63% 12:3 12/15 22/22 29/2/4 29/2 13:63 25:30 12/12 6:20 12/22 6:00

**Fourth Step: the User Controls but the System Helps (a Lot).**

Before speaking about formulas, generation of series is a good example of how the system helps the user (learner). What happens when expanding down the cells B2, C2..., F2 and what about expanding the blocks G2..G3, H2..H3 and I2..I3?

<table>
<thead>
<tr>
<th></th>
<th>A</th>
<th>B</th>
<th>C</th>
<th>D</th>
<th>E</th>
<th>F</th>
<th>G</th>
<th>H</th>
<th>I</th>
<th>J</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td></td>
<td></td>
<td>Tips</td>
<td>Week</td>
<td>March</td>
<td>2000</td>
<td>John</td>
<td>Lennon</td>
<td>06:00</td>
<td>06:00</td>
</tr>
<tr>
<td>2</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>1</td>
<td>1.5</td>
</tr>
<tr>
<td>3</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>2</td>
<td>1.8</td>
</tr>
</tbody>
</table>

Good formatting of a spreadsheet takes these considerations into account.

**Fifth Step: Formulas Are the Core Business of the Spreadsheet Program.**

The first examples and exercises focus on the identification of calculated values. Formulas do not include functions. The table is adapted to the discipline because it is related to pre- and post-tests. At this stage, the (re)copy feature is not necessary but may be used.

<table>
<thead>
<tr>
<th></th>
<th>A</th>
<th>B</th>
<th>C</th>
<th>D</th>
<th>E</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>PRE</td>
<td>POST</td>
<td>BENEFIT</td>
<td>R. B.</td>
<td></td>
</tr>
<tr>
<td>2</td>
<td>Vocabulary</td>
<td>35%</td>
<td>70%</td>
<td>37%</td>
<td>56%</td>
</tr>
<tr>
<td>3</td>
<td>Grammar</td>
<td>65%</td>
<td>85%</td>
<td>25%</td>
<td>52%</td>
</tr>
</tbody>
</table>

**Sixth Step: (re)Copy Is a Basic Tool for the Good Formatting of a Spreadsheet.**

For instance, how many different formulas are needed to build this table?

<table>
<thead>
<tr>
<th></th>
<th>M</th>
<th>N</th>
<th>O</th>
<th>P</th>
<th>Q</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>6</td>
<td>9</td>
<td>8</td>
<td>2</td>
<td></td>
</tr>
<tr>
<td>5</td>
<td>30</td>
<td>46</td>
<td>40</td>
<td>10</td>
<td></td>
</tr>
<tr>
<td>7</td>
<td>42</td>
<td>63</td>
<td>56</td>
<td>14</td>
<td></td>
</tr>
<tr>
<td>9</td>
<td>54</td>
<td>81</td>
<td>72</td>
<td>16</td>
<td></td>
</tr>
<tr>
<td>3</td>
<td>18</td>
<td>27</td>
<td>24</td>
<td>6</td>
<td></td>
</tr>
</tbody>
</table>

Just one. And which one?

**Seventh Step: Other Basic Tools Are Cells Blocks Formatting and Conditional Formatting.**

The following example shows that particular exercises may motivate learners and help
them learning more and by themselves. In that case, an exercise makes sense, even if it is not directly related to the learner's discipline.

<table>
<thead>
<tr>
<th></th>
<th>A</th>
<th>B</th>
<th>C</th>
<th>D</th>
<th>E</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>2</td>
<td>apple</td>
<td>pear</td>
<td>banana</td>
<td>cherry</td>
<td></td>
</tr>
<tr>
<td>3</td>
<td>grapefruit</td>
<td>apricot</td>
<td>pineapple</td>
<td>hazelnut</td>
<td></td>
</tr>
<tr>
<td>4</td>
<td>melon</td>
<td>orange</td>
<td>lemon</td>
<td>watermelon</td>
<td></td>
</tr>
<tr>
<td>5</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Any fruit name, whatever it is, written in this table, must be displayed in red if the name is more than six characters long.

Students like fruits but also this kind of exercise.

**Eighth Step: the Spreadsheet Program Is Helpful When Data Are Numerous.**

This step is probably a bit more specific to the audience. Surveys they have to do often concern lots of people, lots of questions or items... To teach them how to modify or simply consult large tables (hiding rows or columns, using elaborated filters or not...) is certainly a mean to increase their efficiency.

**Ninth Step: Data Sometimes Already Exist Elsewhere.**

This step is also quite specific to the audience. It helps increase efficiency. As data often exist elsewhere (in another sheet created by another spreadsheet program, in a table of a text document or if they are structured in a text document...) they may easily be imported. Notice that, more generally, exercises about these features prevent students from losing time encoding and allow them to focus on the important thing. So, for instance, they can work on large data sets (see previous steps).

**Tenth Step: the System Easily Generates Charts but, Designing a Good Chart Is not so Easy.**

Concepts about charts are limited. Learners quickly discover how to ask the system to display a chart. In this case, sobriety is often missing. But choosing a relevant type, avoiding redundancy, choosing scales in order to provide a good idea of what data should show... are abilities that should be promoted.

4 CONCLUSION

It is not so obvious to point out what is important when teaching about ICT. The reasons are that the experience is short, that disciplines are young (even if spreadsheet programs have been existing for years). There is also a big temptation to focus on products and concrete results rather than on real abilities and competences to develop among learners. In this paper, we suggested a method to elicit fundamental knowledge related to the spreadsheet program. This method is based on the common existing features of all the spreadsheet programs and the concepts they convey. We named these common features “principles”. And we named principles and concepts “invariants”.

Invariants help us to define a trail to discover the power of the spreadsheet programs, but
they are also means, a way of thinking that helps learners to go further by themselves. Notice that invariants may be identified for each software package, not only for the spreadsheet program.

Once knowledge elicited, the design of the trail will depend on the audience. Here, we incompletely illustrated an example with some flashes on a particular trail, trying to precise some of its specific objectives. A few examples were given to suggest some recommendations like providing illustrations and exercises, that…

• focus on the main concept(s) (i.e. put the challenge on what has to be learned);
• promote exploration (i.e. use of a particular function, discovery of how the system works);
• excite learners’ motivation through challenges.

All this looks very simple. But choosing good examples, creating interesting exercises, respecting the three previous recommendations take time. If you find such examples, keep them carefully and better, try to share them with others. Books about spreadsheets are full of illustrations showing sales tables, bills and pay slips or whatever, but examples like those we showed remain unfortunately rare.

5 REFERENCES


This page intentionally blank
Spreadsheets and the Financial Collapse

Grenville J. Croll
Chair, European Spreadsheet Risks Interest Group
www.eusprig.org
Grenville@spreadsheetsrisks.com

ABSTRACT

We briefly review the well-known risks, weaknesses and limitations of spreadsheets and then introduce some more. We review and slightly extend our previous work on the importance and criticality of spreadsheets in the City of London, introducing the notions of ubiquity, centrality, legality and contagion. We identify the sector of the financial market that we believed in 2005 to be highly dependant on the use of spreadsheets and relate this to its recent catastrophic financial performance. We outline the rôle of spreadsheets in the collapse of the Jamaican banking system in the late 1990’s and then review the UK financial regulator’s knowledge of the risks of spreadsheets in the contemporary financial system. We summarise the available evidence and suggest that there is a link between the use of spreadsheets and the recent collapse of the global financial system. We provide governments and regulating authorities with some simple recommendations to reduce the risks of continued overdependence on unreliable spreadsheets. We conclude with three fundamental lessons from a century of human error research.

1 INTRODUCTION

Four years ago [Croll, 2005], we interviewed anonymously 23 senior individuals to determine their views regarding the importance and criticality of spreadsheets in the City of London, where they all worked. The details of our motivation, methodology, selection of participants, method of data collection and the background and structure of the interviews are given in that earlier paper. Interviewees provided much useful information, which was best summarized by a financial regulator:

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

This succinct comment was reviewed and validated by a senior investment research professional working in a first tier investment bank who added:

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

Based upon the evidence provided by the interviewees, and the historical evidence of the weaknesses and limitations of spreadsheets, we concluded our article thus:

“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”

At the time, this conclusion was felt to be rather bold, however following distribution of the paper to the interviewees and during the peer review process, no adverse comments arose and the text was left as written. We are thus in the position of having:

i) An extensive spreadsheets risks research base which identifies spreadsheets as a considerable weakness in organisations
ii) Research evidence outlining the ubiquity, importance and criticality of spreadsheets in the City of London
iii) An *a priori* hypothesis about catastrophic financial loss due the use of untested spreadsheets in the City of London and
iv) A systemically collapsed financial system deeply affecting the City of London

Our present research hypothesis, an extension of iii) above, is to determine by a methodical examination of the evidence available to us whether or not spreadsheets played a rôle in the collapse of the global financial system.

We now review the spreadsheet risks research base and our previous work in more detail, extending the work slightly by adding some email and other evidence that was omitted. We then examine the rôle played by spreadsheets in the collapse of the Jamaican banking system in the late 1990’s. Following this, we outline the UK Financial Regulator’s knowledge of the problems with spreadsheets by reviewing and summarising a series of public presentations on the issue. We cover briefly the fiduciary duty of UK company directors and their compliance in light of their permitting the extensive use of untested and uncontrolled spreadsheets in financial recording and reporting. Following our conclusions we repeat and then extend a set of simple recommendations.

2. SPREADSHEET RISKS RESEARCH

There is a very extensive research base on the risks of using spreadsheets within business [Panko, 2000][Panko & Ordway, 2005][Powell, Baker & Lawson, 2008]. Much of the research has been coordinated and progressed by the European Spreadsheet Risks Interest Group [Chadwick, 2003][EuSpRIG, 2009]. Further significant work improving the end user approach to software has been undertaken by the EUSES consortium [EUSES, 2009]. The main known risks of spreadsheets include:

a) Human Error – To err is human, hence the majority (>90%) of spreadsheets contain errors. Because spreadsheets are rarely tested [Panko, 2006][Pryor, 2004] these errors remain. Recent research has shown that about 50% of spreadsheet models used operationally in large businesses have material defects [Powell, Baker, Lawson, 2007] [Croll, 2008]. Approximately 50% of executives recently surveyed had encountered spreadsheet related problems up to and including staff dismissal [Caulkins, Morrison & Weideman, 2007].

b) Fraud – Because of the ease with which program code and data is mixed, spreadsheets are the perfect environment for perpetrating fraud [Mittermeir,
examination of the evidence available to us whether or not spreadsheets played a rôle in the collapse of the financial system.

We now review the spreadsheet risks research base and our previous work in more detail, in light of their permitting the extensive use of untested and uncontrolled spreadsheets in the City of London and systemic banking collapse in a later section.

Unfolding events suggest that there are several further categories of risks related to spreadsheets that have not been studied at all:

f) Assumptions – Most spreadsheet models rely upon a fairly lengthy series of explicit or more usually, implicit, assumptions. Not least of these is the *Ceteris Paribus* assumption of all other things being (or remaining) equal. Clearly the assumptions underlying many spreadsheet models now no longer apply.

g) Opacity – Initial use of a spreadsheet gives a feeling of transparency, where any and all aspects of the model can be clearly addressed and viewed. With larger models, more complex models, badly structured models or models that have been developed by somebody else, the spreadsheet becomes opaque and unwieldy.

h) Reification – Once information is put into a spreadsheet, it becomes concrete: "...things changed with the intrusion of the spreadsheet. When you put an Excel spreadsheet into computer-literate hands you get a 'sales projection' effortlessly extending ad infinitum! Once on paper or on a computer screen .....the projection takes on a life of its own, losing its vagueness and abstraction and becoming what philosophers call reified, invested with concreteness; it takes on a new life as a tangible object" [Taleb, 2007]

i) Enterprise Interoperability – The interconnected heterogeneous nature of the financial system and the spreadsheets contained within it require us to recognise that we can no longer continue with the closed world deterministic assumptions of traditional IT: "...for with globalisation we are no longer concerned with intra-operability contained within closed systems but with inter-operability between open systems. The closed world assumption of Gödel completeness has been adequate for the logic of closed systems but his undecidable theorems show that the logic of open systems needs to go beyond a mathematics of axioms, sets and number". [Heather & Rossiter, 2008]. Heather recently expressed the view that this issue underpinned the present crisis [Heather, 2008].

The above points comprise a high level taxonomy of spreadsheet risks. There has been substantive previous work on taxonomies of spreadsheet errors [Rajalingham et al, 2000][Panko, 2008]. We note that previous taxonomies do not address issues relating to the
risks involved in the use of large systems of interlinked spreadsheets or some of the other human factors identified above.

3. THE RÔLE OF SPREADSHEETS IN THE FINANCIAL SYSTEM

Our previous paper [Croll, 2005] gave a reasonably comprehensive view of how spreadsheets are used in the City of London. More recent direct personal experience in the Enterprise Spreadsheet Management System marketplace [Baxter, 2007] [Saadat, 2007] [Perry, 2008] within the City of London and other global financial centres provided further useful information. We did not find any evidence which contradicted that reported previously. We summarise the various themes developed in our earlier paper, relating them to current events and adding several other aspects.

3.1 Ubiquity

Amongst a number of observations highlighting the truly ubiquitous use of spreadsheets in the City of London, we made an observation that:

“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”

We note that Collateralised Debt Obligations (CDO’s), are a popular form of credit derivative. CDO’s offer a structured payoff depending upon default levels of underlying assets (including further CDO’s) and have played a significant part in the present financial crisis, earning the sobriquet “Toxic Debt”. There is good evidence that CDO’s are valued within complex spreadsheets:

“Standard & Poor’s CDO Evaluator Handbook Version 3.0. An Introduction to and operational overview of Standard & Poor’s Evaluator Tool, an automated spreadsheet application used for rapid, complex evaluation of the credit quality of CDO asset portfolios”. [Standard & Poor’s, 2006]

Note that we have not examined this spreadsheet, only the published documentation, in the manner of our analysis of some spreadsheet documentation used in clinical medicine [Croll & Butler, 2006].

Recently, the UK operation of a large international investment bank was fined £5.6m for breaches of several regulatory principles by the Financial Services Authority:

“2.33.3. The booking structure relied upon by the UK operations of Credit Suisse for the CDO trading business was complex and overly reliant on large spreadsheets with multiple entries. This resulted in a lack of transparency and inhibited the effective supervision, risk management and control of the SCG {Structured Credit Group}” [FSA, 2008].

The problems with CDO’s were first identified in 2007, however the full system collapse did not start until a year later:
“....Merrill Lynch & Co.'s threat to sell $800 million of mortgage securities seized from Bear Stearns Cos. hedge funds is sending shudders across Wall Street. A sale would give banks, brokerages and investors the one thing they want to avoid: a real price on the bonds in the fund that could serve as a benchmark. The securities are known as collateralised debt obligations, which exceed $1 trillion and comprise the fastest-growing part of the bond market. Because there is little trading in the securities, prices may not reflect the highest rate of mortgage delinquencies in 13 years. An auction that confirms concerns that CDOs are overvalued may spark a chain reaction of write downs that causes billions of dollars in losses for everyone from hedge funds to pension funds to foreign banks. Bear Stearns, the second-biggest mortgage bond underwriter, also is the biggest broker to hedge funds.” [Pittiman, 2007]

Note that since there was very little trading in CDO’s, the financial valuations expressed in CDO spreadsheet models were reified in the manner summarised earlier by Taleb.

Note also that spreadsheet use in the Peoples Republic of China may be less ubiquitous than in the West [Turner, 1995] and may provide a useful contrast in future research.

Our previous observations about the ubiquity of spreadsheets and their heavy use in the more recent credit derivatives marketplace are therefore well founded. Complex financial instruments such as CDO’s are implemented within, and valued by, large and complex spreadsheets. At least one large firm has been fined by the financial regulator for failure to control these spreadsheets. CDO’s and other credit derivatives played a leading rôle in the collapse of the global financial system.

3.2 Importance

Recent discussions with the senior management teams at a variety of financial institutions revealed that spreadsheets continue to play a rôle at least as important as that depicted in our earlier paper. We give some examples, though there are very many more:

a) We were asked to investigate a spreadsheet belonging to a top investment fund that was a member of the Investment Management Association (IMA). The spreadsheet entitled (perhaps) BALDEC08.XLS contained the profit & loss statement & balance sheet of the fund for 31 Dec 08. The Net Asset Value closely matched that on the IMA website. This large & complex spreadsheet had not undergone any formal testing or approval and was undocumented [Payette, 2006], unprotected and uncontrolled.

b) One very large institution with over 750 business critical spreadsheets in use in the finance and other departments, recently employed in excess of 30 specialist contractors for a substantial period to remediate and validate these spreadsheets prior to placing them under a formal system of control.

c) A large bank was using a suite of several hundred spreadsheets to underpin the operations of one of its key lending departments. There was no evidence that any formal testing of these spreadsheets had taken place and at the time of writing they were uncontrolled.
d) A trading desk of a large foreign bank used spreadsheets as the basis for its derivatives operations.

The firm in the second example above was one of only a handful of UK organisations known to be systematically and professionally addressing business critical spreadsheets used in financial reporting and other important processes. By contrast, we became aware of some large organisations who were simply placing, or intending to place, untested spreadsheet applications under formal control, without testing or validating them first. One or two very large firms had set up manual spreadsheet control systems, which are known to fall into disrepair within a year or so. One FTSE 100 organisation denied the use of spreadsheets for anything other than innocuous purposes. The denial was less than convincing.

It would appear to be the case that there are presently about 25 departmental installations of Enterprise Spreadsheet Management systems within the City of London. That is, a small fraction of the institutions have begun to acknowledge the importance of spreadsheets and have asserted a need to discover, risk assess, test and control them. Most of these installations are within institutions that became insolvent and were subsequently bailed out by the authorities. It would appear to be the case, at present, that the overwhelming majority of key & critical spreadsheets used within UK based (and by implication, global) financial institutions are completely uncontrolled and that about 50% of them are also untested.

3.3 Criticality

We previously defined a critical spreadsheet as one where:

“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”

The large number of spreadsheets containing the details and valuations of myriad CDO’s and other complex financial instruments have clearly compromised governments, regulators and financial markets. As the magnitude of the present financial crisis has become clear, fingers of blame are beginning to search for causes and those responsible. We believe that spreadsheets, and those who used them, or permitted their use, in ignorance of the well-known risks [e.g. Miller, 2005] and consequences, share much of the blame. Given the risks of using spreadsheets and their ubiquity in financial trading, recording and reporting, it is difficult to see how most organisations comply with the basic tenets of the Companies Act, let alone higher statutes such as Sarbanes Oxley and Basle II [Panko & Ordway, 2005].

3.4 Legality

The question of the fiduciary responsibility of company directors and officers in relation to the use of untested and uncontrolled business critical spreadsheets has been initially addressed [Cleary, Norris-Jones, Madahar, 2005]. The general question of fiduciary duty given the present circumstances, is within the current debate [Price, 2009]. The UK Companies Act is very clear on the requirement that directors shall keep accurate records, as section 221 of the act states that:
Clearly, given that approximately half of spreadsheets used in the operational and financial management of large corporates have been found to be materially defective, it would appear to be the case that, in recent years, many directors of UK based banks and financial institutions may not have fully complied with section 221 of the Companies Act. We suggest that it may indeed be unlawful for companies to continue using untested and uncontrolled spreadsheets in critical parts of the financial recording and recording infrastructure. The issue of undue reliance on (spreadsheet based) financial models has now come before the UK courts [Herman, 2008]

Moreover, in our view, fiduciary responsibility in respect of the use and misuse of spreadsheets lies both sides of the regulatory divide.

3.4 Centrality

The FSA’s Gabriel and Gabriella [FSA, 2009] systems for collecting periodic (i.e. daily, weekly, monthly & annual) statutory financial information from regulated institutions are spreadsheet based. The Bank of England likewise routinely collects financial information from across the markets on spreadsheets. The daily cash reconciliations performed by the clearing banks after the 15:30 daily close are performed on spreadsheets [BCS, 2000]. There exists within the City of London a critical spreadsheet called LIBOR.XLS. The capital adequacy ratios of the institutions were known to be stored and monitored from within a regulator’s spreadsheet.

The regulating authorities use spreadsheets as intensively as market participants, but in this case spreadsheets are used for the purposes of monitoring, regulating and forecasting the markets themselves. There is little evidence that regulators have put in place control processes to manage the spreadsheets they use to manage the financial system.

3.5 Contagion

Back in 2005, the anonymous UK regulatory interviewee clearly understood the relationship between Value at Risk (VAR), Tail Risk, Regulatory Capital & Financial Contagion. The following email was omitted from our earlier paper:

“...it might be useful to give a sense of how much capital is at risk. Institutions measure market risk by VAR [Value at Risk], which is to a 99% confidence level, daily horizon. Capital is set at between 10 and 20 times that number. ‘Tail risk’ is the risk of extreme events, where all capital held by a firm is lost (and where potentially it is forced into default). This could be 40 times VAR or more, i.e. multiples of the capital..."
buffer, and when this gets to such high levels, this has a ‘contagion effect’ – the firm
cannot pay creditors out of capital, the creditors also realise losses, perhaps they too
are forced into insolvency, thus forcing other firms, and so on. The total amount at
risk in the global system is massive (obviously). How much of this is managed on
spreadsheets? We don't know for sure, but we have specific examples where this
happens. We also have a number of concrete examples where the spreadsheets (one
due to a complex cell formula) failed. In one case a loss was realised. In another,
there was a significant period when the risk was being mismanaged – fortunately this
had the effect of reducing the risk, rather than increasing it. This risk impacts one of
the FSA's 4 core objectives, which is market stability". [FSA, 2005a]

The email was omitted because there was then no clear or definite link between spreadsheets
and financial contagion. Plus of course, the risks of contagion were perceived to be small at
the time. The fact that the UK regulator raised the issue of the relationship between financial
contagion and spreadsheets is now an interesting point.

The author is aware of large spreadsheets incorporating monte carlo simulation techniques
having been in widespread use in the past to calculate the Value at Risk of large firms. The
UK regulator is also more recently aware of such use, expressing certain technical concerns:

"We have come across some firms that perform Monte Carlo simulations for capital
calculations using standard personal computer spreadsheet...this may prove
inadequate when making a robust assessment of capital...". [FSA, 2008a]

The degree to which it has been possible to calculate daily VAR on assets & liabilities where
the primary transaction record is a spreadsheet is not known.

4 THE RÔLE OF SPREADSHEETS IN THE COLLAPSE OF THE JAMAICAN
BANKING SYSTEM

The Jamaican Banking System collapsed in its entirety in the late 1990’s partly due to the use
of spreadsheets and a consequent failure to manage and control them [Lemieux, 2002, 2005].

The Jamaican banks relied heavily on spreadsheets because their major transaction
processing and risk management systems failed to meet their management information and
reporting requirements. Former employees of the failed banks revealed in later interviews
that they had used spreadsheets in the following ways:

- Cash management
- Financial budgeting and control
- Analysis of product and customer profitability
- Analysis of the cost of capital
- Foreign Exchange management
- Credit decision management
- Interest rate sensitivity analysis
- Financial risk management
- Recording of proprietary transactions
The pattern of usage of spreadsheets in Jamaica in the late 1990’s as outlined above is a small subset of the present pattern of usage of spreadsheets in the City of London and other major financial centres. The particular problems that gave rise to the Jamaican financial collapse were clear:

- Individualistic naming of files. These idiosyncratic file names obscured the content and purpose of the files, divorced the files from the business process and the people creating them (particularly if an individual had left the bank), prevented easy access and made management reporting very difficult.

- Ad hoc assignment of storage location. Important files were stored on personal hard drives without any document management, backup or archiving.

- No Document Retention policy. Data was deleted at the whim of the individual so that banks financial positions could not be calculated accurately or even at all.

- Failure to preserve a link to the business context. This rendered spreadsheets meaningless as a background to a particular business decision.

- Inability to guarantee the authenticity and reliability of spreadsheets. 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. As a result, 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.

The problems with spreadsheets caused numerous problems, including:

"Poor control over spreadsheets at Jamaican indigenous banks contributed to management information and external reporting problems (i.e., P&L distortions) that 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”[Lemieux, 2005]

and

"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”.[Lemieux, 2005]

It is highly likely that these limitations in spreadsheet management and control experienced in Jamaica during the late 1990’s are now being experienced in the wider, global economy.
5. UK REGULATOR’S KNOWLEDGE OF SPREADSHEET RISK


The first presentation “[End] User Computing in Financial Regulation”, outlined the FSA’s approach. The FSA did not view End User Computing (EUC) as a bad thing, as it was essential to business efficiency:

“The FSA is not, and never has been opposed to the use of spreadsheet and other user-developed applications for business critical purposes”. [FSA, 2003]

It was clear however that the regulator expected appropriate controls to be in place. At that time it was clear that the senior management (of regulated firms) thought that end user computing was “bad” and sought to replace it with a “strategic” IT solution. Development of strategic solutions was hampered by the “budget paradox”. Generally, it was impossible to find a budget for any form of IT development required by the business, which implied that the firm could not afford it. However, it was always the case that some salaried employee of the firm could find the time, and a non-IT budget, to develop the required solution, which implied that the firm could afford it. This inevitably led to poorly managed solutions. It was clear to the FSA that EUC solutions were cheap to develop, but that the resulting disasters were not cheap. A change of mindset was called for, with attention being paid to training, standards and the traditional IT concerns of auditability, testability and maintainability. At the time, the view was that IT auditors focussed their attention on large information systems and tended to regard spreadsheets as being user problems, not of their concern.

The second presentation “Why Banks use Spreadsheets” sought to distinguish business critical applications from non business critical applications, identified the typical uses of spreadsheets illustrated by two case studies and concluded with a look of the cost of controls. The regulators position on the use of spreadsheets within business critical systems was clear:

“It is crucial to distinguish so-called ‘business critical’ systems from all others. Financial regulators are primarily worried, for obvious reasons, about systems which are crucial to the successful and safe running of businesses. For example, systems which support the provisioning of liquidity to the money markets, which support global foreign exchange platforms, [or] which control the management of financial risk. This talk is concerned only with ‘business critical’ systems. I will talk about how many banks use business critical systems implemented on spreadsheets, based on evidence from many years of supervising banking systems”. [FSA, 2004]

The regulator defined a business critical system as any system whose function is critical to the running of a business. Business critical systems were systems which: have more than one user; where the user probably doesn’t know the “right answer”; where they were performing a cyclical or repetitive task; where there was usually data processing of some kind going on such as aggregation, enrichment, checking or MIS. Note that this definition of criticality differs from our own definition. Furthermore:
“Business critical applications are typically developed out of ‘one off’ spreadsheets which were not thrown away because they were seen to perform a useful routine function. Because the function was so routine and boring, they ‘give’ it to some less highly paid individual who does not understand what the spreadsheet is doing, with some instructions that s/he is able to follow only on a highly literal basis. [The] operator has little incentive to get the answer right apart from the fact they are paid to do it” [FSA, 2004]

and

“Financial regulators see a lot of business critical applications developed on spreadsheets. They worry a lot about them, in case (a) they go wrong and the firm loses a lot of money in a way that disrupts the operation of the financial system. This has happened before, though not because of spreadsheet systems (so far) [and] (b) someone exploits the inherent security vulnerability of spreadsheets to commit some bad crime. This HAS happened before.” [FSA, 2004]

The reasons why end users build systems, rather than wait for IT to build them were clearly articulated:

“User developed systems, by contrast are built by small teams of people who understand exactly their own requirements, and are motivated to build exactly what they require in order to achieve their primary goal, which is a tool to further their business. Every minute they spend on the system (for them) is wasted effort, because they want to do something else. They only build the system because, without it, they cannot do what they actually want. Thus they have a strong incentive to minimise costs, and have by and large no reason to incur those costs in the first place.” [FSA, 2004]

The third and fourth presentations, in 2005 and 2007 both entitled “Regulatory Update” reviewed the progress on the Spreadsheet issue within regulated institutions over the preceding several years.

In the regulators view, the management mind set had changed as spreadsheets were increasingly accepted as strategic solutions, which was a major change from five years previously. Many firms had given up on the ‘big solution’ route and accepted end user computing for what it was. The regulator noted the increasing use of spreadsheet control solutions, which removed the burden of compliance from the end user.

The regulator noted some good news from the audit profession where they were seeing more mention of spreadsheets in audit reports and that increasingly EUC was part of the audit plan. The Sarbanes Oxley act had assisted in controlling EUC as it required firms to demonstrate that controls around financial reporting are adequate. This requires the auditor to assess the effectiveness of managements’ assessment of internal controls. It was noted that the requirement to demonstrate that effective EUC controls are in place was particularly onerous because EUC applications are usually not documented. Most problems however, were still the result of poor use of EUC solutions. Good training was an obvious solution, but there was little budget.
6. CONCLUSIONS

We have shown in previous sections and the literature supporting them that:

a) there is a large research base extending back over twenty years demonstrating that spreadsheets pose significant risks to organisations
b) spreadsheets are integral to the function and operation of the global financial system and most users of the financial system do not realise this
c) catastrophic collapse of one form or another due to the use of untested spreadsheets in the financial markets was predicted several years ago
d) a recent previous regional banking collapse was partly caused by the use and abuse of spreadsheets
e) the UK financial regulator was fully cognisant of the risks of using spreadsheets within the banking and other regulated financial sectors, the related possibility of contagion and relevant technical problems in the use of spreadsheet based Monte Carlo techniques to compute VAR based capital requirements
f) credit derivatives were identified as being particularly reliant on spreadsheets
g) a systemic collapse of the global financial system occurred during the period 2007-2009 where credit derivatives played a significant part in the destruction of capital

We have confidence in concluding that spreadsheets played a rôle, perhaps even a significant rôle, in the recent collapse of the financial system, affirming our research hypothesis. In our opinion, their primary rôle is centred around the fact that they were one of the principal technologies used in the Credit Derivatives marketplace. This market grew very quickly due to the ease with which it was possible to design and promulgate opaque financial instruments based on large spreadsheets with critical flaws in their key assumptions. Their secondary rôle is centred around their ubiquitous use such that many of the problems experienced in Jamaica – specifically the difficulty in valuing or risk assessing complex financial instruments pre and post crash contributed to the cause of the crash and the difficulties experienced in recovering from it. The tertiary rôle of spreadsheets is related to the issues associated with human error and other human factors, which are no doubt compounding difficulties in market recovery and will remain a problem for the future unless and until resolved.

During the course of the preparation of this article, we have identified a further set of risks associated with spreadsheets (and, by implication, other forms of end user computing). These give rise to the suspicion that the global financial system may be deeply flawed, requiring extensive modification and repair. Despite the recent recapitalisation and part nationalisation of the institutions, most of the spreadsheets still remain. In many cases these spreadsheets will now be operated by people not familiar with them and almost certainly outside of the domain where their assumptions and formulae were hoped to hold true. We are confident in predicting that further obvious problems with the financial system will come to light for some time unless and until untested and controlled spreadsheets play a less significant rôle in the financial system.

The ubiquitous use of uncontrolled spreadsheet technology leaves the financial system open to future systemic problems. Certain parts of the financial system have been homogenised by the ubiquitous use of spreadsheets. Those parts of the financial system are thus exposed to all the risks of spreadsheets, not merely the lack of testing or control. Note that the more
heterogeneous interoperating systems developed using traditional IT methods which implement other parts of the financial system (eg cash transmission & retail banking), did not fail.

Arguably, if there had been a requirement that CDO’s and such like had to be implemented or valued within traditional IT systems, or in spreadsheets using the full software development lifecycle, the CDO may never have seen the light of day.

**7 RECOMMENDATIONS**

Clearly, it is important that the financial system is resilient and stable enough to support the global economy. In our view, given the critical and burgeoning risks of spreadsheets and their ubiquity, this cannot happen without extensive changes in the demographics of their use & management [Powell, Baker, Lawson, 2005]. When we make a journey on public transport by air, rail, road or sea, there is high probability of our safe arrival. The same should apply to our expectations of surviving a financial journey, and much change will be required to achieve this.

Our previous recommendations still apply:

“*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*”. [Croll, 2005]

A very few organisations have voluntarily adopted Enterprise Spreadsheet Management systems, implementing them mainly on a departmental rather than a strategic basis. There is no evidence in the public domain or accessible to us that suggests that these control systems are truly effective during long-term use in the financial sector, though it is our hope and expectation that they are. We need to determine by detailed examination of the systems already in place that spreadsheet control systems do give the requisite degree of long term control to spreadsheet based financial systems, or financial systems where spreadsheets play a significant rôle.

Having determined that it is possible to effectively control spreadsheets, the regulators must compulsorily require that all participants in the global financial system discover, risk assess, test and control the use of spreadsheets already in place as part of the system. Spreadsheets that necessarily must become part of the financial system must go through the full software development lifecycle [Grossman, 2002]. They must be designed, implemented and
controlled by people professionally trained in their creation, testing and deployment. This can only happen through global governmental and regulatory cooperation and may, due to cost, require a reduction in the scale and scope of the regulated financial system.

If it is the case that we cannot build a safe and effective global financial system using spreadsheets as a key component, then we will have no alternative but to gradually replace them, which will be time consuming and expensive, if indeed it is possible at all. The decision to replace a spreadsheet or set of spreadsheets with an alternative is a straightforward commercial decision, influenced by regulatory considerations, which would look at the risks, costs and benefits of each of the alternatives.

Finally, in our attempt to build a better financial system, it is very important to note that a century of research on human error informs us that humans have predictable limitations in many cognitive domains, including spreadsheets [Panko, 2007]. These limitations must be taken into consideration as we make decisions about technologies including spreadsheets. Panko’s conclusions provide excellent strategic guidance, which we would be unwise to ignore:

“It is fundamentally important to understand that the problem is not spreadsheets. Asking what is wrong with spreadsheets has some value, but it is fundamentally the wrong issue. The real issue is that thinking is bad. Of course, thinking has some benefits, and we are accurate 96% to 98% of the time when we do think…..The second fundamental thing to understand from human error research is that making large spreadsheets error free is theoretically and practically impossible….. {and to attempt to do so} is a testament to the profound human tendency to overestimate their ability to control their environment even when they cannot. The third fundamental thing to understand is that replacing spreadsheets with packages does not eliminate errors and many not even reduce them. The problem, again, is that thinking is bad. Unless there are no human decisions or design actions in using a package, there will be thinking, and therefore there will be errors” [Panko, 2007]

We nevertheless hope that this paper has set out many of the hitherto largely unknown risks of the use of spreadsheets in the global financial system such that better informed decisions regarding their future use can be made. Whatever technology choices are made for the future, human factors [Thorne & Ball, 2005] will remain the overarching consideration.

8 ACKNOWLEDGEMENTS

The author thanks a number of EuSpRIG colleagues for comments on an early draft of this paper. This work was supported by the author during the period March-May 2009. There are no conflicts of interest to declare.

9 REFERENCES

http://arxiv.org/abs/0801.1856
controlled by people professionally trained in their creation, testing and deployment. This can require a reduction in the scale and scope of the regulated financial system. Finally, in our attempt to build a better financial system, it is very important to note that a straightforward commercial decision, influenced by regulatory considerations, which would ignore:

Panko’s conclusions provide excellent strategic guidance, which we would be unwise to ignore:

“...the profound human tendency to overestimate their ability to control their environment even when they cannot...”

The third fundamental thing to understand is that replacing spreadsheets with packages does not eliminate errors and benefits, and we are accurate 96% to 98% of the time when we do think...The second wrong issue. The real issue is that thinking is bad. Of course, thinking has some value, but it is fundamentally the...
Heather, M., (2008), Private communication, 30th Meeting of the Alternative Natural Philosophy Association (ANPA), Wesley College, Jesus Lane, Cambridge


This page intentionally blank
Reducing the Risk of Spreadsheet Usage  
– a Case Study

Mel Glass (EASA), David Ford (Amlin), Sebastian Dewhurst (EASA)  
Davenport House, 39 Evenlode Drive  
Wallingford, Oxfordshire, OX10 7NZ, UK.  
info@easasoftware.com

ABSTRACT

The frequency with which spreadsheets are used and the associated risk is well known. Many tools and techniques have been developed which help reduce risks associate with creating and maintaining spreadsheet. However, little consideration has been given to reducing the risks of routine usage by the ‘consumers’ - for example when entering and editing data. EASA's solution, available commercially, ensures that any routine process involving spreadsheets can be executed rapidly and without errors by the end-users, often with a significant reduction in manual effort. Specifically, the technology enables the rapid creation and deployment of web-based applications, connected to one or more centralized spreadsheets; this ensures version control, easy and error-free usage, and security of intellectual property contained in spreadsheets.

1. INTRODUCTION

Countless surveys have revealed the ubiquitous nature of spreadsheets; it has even been suggested that spreadsheets are a universal technology [Durfee, 2004].

This is especially true in the financial sector, due in part to the relative extremes of specialisation within the industry [Sentence, 2006]. Specialisation of end-user knowledge, coupled with short delivery timeframes, makes it difficult to deliver custom solutions using conventional IT technologies. It is simply quicker for end-users to build their own tools with spreadsheets than it is to explain to an IT professional what kind of application is required, and then wait for it to be built.

This, of course, has given rise to the recognition of the serious lack of spreadsheet control [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].

Methodologies and commercial products intended to identify errors in spreadsheets during the creation, testing, and subsequent maintenance are becoming more commonplace [Ayalew et al, 2000; Bekenn, 2008]. However, little consideration has been given to control of the spreadsheets as they are used by the consumers who use these spreadsheets to execute routine processes. There are some notable exceptions [Chambers & Hamill, 2008], and in fact some have considered the Excel “supply chain” [Samar & Patni, 2005]. The all-to-frequent result of this is that, if the problem is addressed at all, extreme measures are implemented in an effort to control usage; for example, we see companies who prevent changes to “approved” spreadsheets by simply locking the spreadsheet.
beyond input values, which is moderately effective but which also has limitations [Baxter, 2007].

We also see the wholesale replacement of operational spreadsheets with custom applications. While this is a preferred route, it is usually expensive and time consuming; therefore spreadsheets will always fill the void between what a business needs today and the formal installed systems, as noted by PWC [Baxter, 2007].

2. AN ALTERNATIVE APPROACH

Our approach, available as a commercially software package called EASA (Enterprise Accessible Software Applications), is a tool for building Web-based applications. These can run existing spreadsheets (and other software) on a Web server, thereby improving usability, accessibility, and control.

2.1 Overview of EASA

EASA has been in use since 2002, by industries as diverse as health-care, communications, energy, financial services, and manufacturing. Specific uses have included the provision of custom interfaces to multiple existing applications, giving users simplified access to key software tools and data [Kornfein & Rajiv, 2008]. Another common purpose is the modernization of legacy software, which can be “wrapped” by EASA and transformed into modern, web-enabled applications, accessible to any desktop, laptop, or mobile device in the enterprise [Casanova, 2003]. EASA’s architecture is shown in Figure 1.

![EASA Architecture Diagram]

**Figure 1. EASA architecture**

However, the area of present interest is the application of the EASA product to spreadsheets; this approach allows the creation of simple web based applications (or EASAPs) that link to one or more key spreadsheets, databases, and other existing
software. This not only eliminates the need to distribute key spreadsheets, but also ensures that they are used precisely the way their authors intended, preventing almost all user-errors. A typical EASAP is shown in Figure 2.

![Figure 2. A typical EASA application, or “EASAP”, which connects to a central spreadsheet](image)

A side benefit includes the ability to link spreadsheets with other enterprise software, and to provide user-specific views – only relevant information is exposed to users. Finally, it is possible to establish a record of who did what, with which spreadsheet, and when.

The results pages provide an option to share reports on completed work with other users; their content is defined by the author, and can contain text, tables, charts, images or even animations generated by the application. Figure 3 shows a typical example.
2.2 What is EASA?

EASA permits the rapid creation and deployment of custom, web-enabled tools without the need for skilled programmers, helping organizations save time and money by improving the efficiency of their processes. EASA applications, or EASAPs, can be deployed over a corporate intranet or over the internet.

A custom EASAP (EASA Application) can drive multiple underlying software tools installed on existing systems throughout the enterprise. The underlying software may be anything from complex “expert only” applications, to legacy systems, modern databases, and spreadsheets.

Each EASAP is built for a specific need within the organization. EASAPs are available over the intranet to authorized users throughout the enterprise, providing secure, simplified access to the company's processes, best practices, expertise and software assets.

EASA's codeless application builder is used to create a user interface, to link the interface to underlying software and databases, and if required, to create custom reports. This eliminates complicated and time-consuming coding of custom applications with tools such as C++, VB, and Java and their associated Integrated Development Environments, which might take weeks, months, or even longer. By comparison, EASA allows new custom applications to be created, tested, and deployed in as little as a few hours.

2.3 How does EASA work?

The EASA software is a client-server environment, in which the information needed to run a particular software application is locked into an EASAP by the author. An EASAP can drive any batch-capable or COM-compliant software (and has dedicated wizards for
linking to Excel), and also allows interaction with ODBC compliant databases. EASA contains the necessary structures to run applications on different computers (where they are resident), to control queuing and user access, and to serve up web-pages to the users.

2.4 How does the Author create an EASAP?

The authoring tools are a particularly powerful part of the EASA system, and a typical EASAP can be created in 2 to 8 hours. EASAP Builder provides a tree structure for building the EASAP and linking it to a centrally maintained spreadsheet, databases and/or other software assets. No knowledge of programming is needed. Figure 4 shows an example of an author’s tree for a typical EASAP.

![EASAP Builder – typical screen](image)

EASAP Builder allows an author to create:
(a) user-interface components such as tabbed panes, choice lists, radio buttons, and input fields;
(b) components that validate data typed into the input fields;
(c) components that perform calculations from this by passing it to a spreadsheet (or other programs);
(d) user-interface components that display results.

EASA also provides a version control system for the EASAPS – allowing them to be developed “off line” by an Author, then, when ready, to publish them on the Intranet. Any subsequent versions are then automatically given revision numbers. Previous versions of EASAPS can be restored by the system administrator and EASAPs can be automatically upgraded to run with the latest version of EASA.
2.5 Key differences between EASA and Sharepoint’s Excel Services

EASA is not the only software for running spreadsheets over the Web: one can also do so with Microsoft's Excel Server. However, Excel Server has two drawbacks:

Firstly, the user experiences an interface that must look like a spreadsheet, while using EASA, an author can design a more intuitive interface, so that users are less likely to make mistakes.

Secondly, if a critical spreadsheet has add-ins or macros, it will not work under Excel Services, while it will under EASA. This is no small issue – we have seen companies list the spreadsheets they wish to deploy, only to find that many hundreds of them have add-ins which would require rework of the order of weeks or even months per spreadsheet. We summarise key differences in Figure 5 below.

<table>
<thead>
<tr>
<th>Feature</th>
<th>MS PerformancePoint</th>
<th>MS Excel ServicesSharePoint</th>
<th>EASA</th>
</tr>
</thead>
<tbody>
<tr>
<td>Tailored productivity tools which create web applications</td>
<td>No</td>
<td>No</td>
<td>Yes</td>
</tr>
<tr>
<td>specific to individual enterprises' requirements</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Database neutral with any SQL database being equally easy to set up.</td>
<td>No</td>
<td>No</td>
<td>Yes</td>
</tr>
<tr>
<td>Including all the internal data which can build up dependency to</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>unwanted database technologies.</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Simple installation and configuration.</td>
<td>No</td>
<td>No</td>
<td>Yes</td>
</tr>
<tr>
<td>Automatic database backing and storage of data to predesigned tables.</td>
<td>Yes</td>
<td>No</td>
<td>Yes</td>
</tr>
<tr>
<td>Codeless custom web application building linked to Excel Service</td>
<td>No</td>
<td>No</td>
<td>Yes</td>
</tr>
<tr>
<td>automatically.</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Data aggregation in database and automatically linked to template sheet</td>
<td>No</td>
<td>No</td>
<td>Yes</td>
</tr>
<tr>
<td>report generation.</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Possible to aggregate and link several spreadsheets in one web application</td>
<td>?</td>
<td>No</td>
<td>Yes</td>
</tr>
<tr>
<td>Externally using Java® Technology with extensions being reusable</td>
<td>No</td>
<td>No</td>
<td>Yes</td>
</tr>
<tr>
<td>between applications and servers.</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>&quot;One version of the truth&quot; design for deployment of spreadsheets to the</td>
<td>No</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td>enterprise.</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Desktop Excel not Required</td>
<td>No</td>
<td>No</td>
<td>Yes</td>
</tr>
<tr>
<td>Excel .xls file can be downloaded with current working data and used</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td>locally for reporting.</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Product neutral.</td>
<td>No</td>
<td>No</td>
<td>Yes</td>
</tr>
<tr>
<td>Avoidance of Excel client configuration for whole enterprise (i.e.</td>
<td>No</td>
<td>No</td>
<td>Yes</td>
</tr>
<tr>
<td>reduced IT overhead).</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Simple and robust security system.</td>
<td>No</td>
<td>No</td>
<td>Yes</td>
</tr>
<tr>
<td>Reusable custom components which can be used and reused by</td>
<td>No</td>
<td>No</td>
<td>Yes</td>
</tr>
<tr>
<td>non-programmers.</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Figure 5. Key differences between EASA and Sharepoint’s Excel Services
3. CASE STUDY

Amlin [www.amlin.com] is a leading independent insurer operating in the Lloyd's, UK, Europe and Bermudian markets. Amlin provides insurance cover to commercial enterprises and reinsurance protection to other insurance companies around the world.

As in any organization, many key processes at Amlin are underpinned by spreadsheets; spreadsheets are easy to create, familiar to users, and flexible. However, lack of control means that the risk of mistakes and the cost of executing the process are unacceptably high.

3.1 Amlin’s RDS application

Staff at Amlin identified one process particularly in need of better control – RDS, or Realistic Disaster Scenario.

RDS requires every agent to submit a six monthly report to Lloyd’s, showing exposure against scenario and by insurance risk code. In Amlin’s case, this means some 20 underwriters around the world must each submit reports.

Historically, this process has been spreadsheet-based; AMLIN underwriters complete and submit a spreadsheet cloned from a template. Complicating this process is the fact that the data required by Lloyd’s changes on a regular basis. Hence a new template must be created, tested and distributed each time there is a change.

The issues which arise include:

- Users change the spreadsheet to suit local requirements before completing and submitting, which results in aggregation errors down-stream.

- The spreadsheet must serve every user, and is therefore more complex than any one user needs – and most users have access to much more of the spreadsheet than is necessary.

- Any change requires that the new spreadsheet is distributed to every user – with instructions to remove the out-dated version. There is no mechanism to ensure the correct version is being used.

- Aggregation is carried out using a series of linked spreadsheets. Links have to be constantly updated, and many layers of spreadsheet refreshed.

- The data exists in many spreadsheets and not in a database, so historical comparison is time-consuming and expensive.

The original spreadsheet is shown in Figure 6.
Figure 6. Before: Amlin’s Realistic Disaster Scenario reporting process required many users to complete and submit a complex Excel spreadsheet. Manual aggregation was time-consuming and error-prone

3.2 Reducing the risk and cost

A solution requiring the complete elimination of Excel was considered. However, that would have required significant investment in building a database application, discarding all the business intelligence already contained in the spreadsheet.

Instead, EASA’s spreadsheet management solution offered a far more cost-effective alternative, allowing Amlin to secure a master version of the spreadsheet on a server. Users now access it only via a custom web application created with EASA’s codeless application builder, allowing a more natural work-flow.

- The custom web application is so intuitive that training is no longer required (see Figure 7).

- Users only see what they need to see; they no longer access the spreadsheet directly, and are not able to “dabble” with it.

- If a change is required, it is made in one place and is immediately published to all users; version control is ensured.

- Aggregation is now automatic

The RDS application is now used in the UK, Bermuda and Singapore. The RDS return for Lloyd’s is now produced at the push of a button, and the process has been error-free since the deployment with EASA.
Figure 7. After: the new web-based RDS application, created with EASA’s codeless application builder, is far easier to deploy and use. It leverages the existing spreadsheet and the business intelligence already embedded in it, eliminates usage errors, and ensures version control.

4. CONCLUSION

A new approach to control the use of critical spreadsheets has been presented, along with a case study demonstrating the effectiveness of the approach. The existing spreadsheets in a company can be leveraged without re-work.

Because this approach facilitates only the deployment and usage of critical spreadsheets, it is complementary to other methods and products which ensure the integrity and auditability of the master spreadsheet itself.

Finally, the technology used allows for integration of critical spreadsheets with other software systems (even legacy applications), and can also be used to provide access to critical spreadsheets for mobile users.
5. REFERENCES


ABSTRACT

"Look before you leap": "a stitch in time saves nine"; "more haste, less speed". Many proverbs declare the wisdom of planning before doing. We suggest how to apply this to Excel, by explaining and specifying spreadsheets before coding them, so there will always be documentation for auditors and maintenance programmers. The specification method uses "pseudo-code": code that, for precision and conciseness, resembles a programming language, but is not executable. (It is, however, based on the notation used by our Excelsior spreadsheet generator, which is executable.) This paper is structured as a tutorial, in which we develop a simple cash-flow and loans spreadsheet.

1. INTRODUCTION

My motivation is stated in the abstract above. If you write your documentation first, and take care to explain what every cell group does, your spreadsheets will be easier to maintain and to audit. Moreover, it is easier to change a design on paper rather than in code.

This paper’s top-level message, therefore, is “document, then implement”. But I am going to suggest a specific way to do this, by using a particular kind of “pseudo-code”. This is the computer science term for notation that looks like a programming language but that is not executable. Like mathematics, such notation is more concise and precise than English, so avoids ambiguities. It's an idea that has been around computer science since at least the 1960s: [Wikipedia] claims it started with the programming language Algol.

You can see this by looking at some examples from Bob Roggio [Roggio]. By using if-then-else statements and loops to describe algorithms, his examples avoid ambiguities such as — in instructions on a shampoo bottle — "wash hair, rinse, and repeat." It isn't clear whether the "repeat" applies to both the washing and the rinsing, or only to the rinsing.

The pseudo-code I describe here is my own design, and rather different from those examples. It is, in fact, based on the spreadsheet-description notation I use in my Excelsior spreadsheet generator [Paine 2009b, Paine 2009c]. But its purpose is the same. Also, it simplifies the job of writing by giving you a fixed pattern to follow.

1.1 Content of this paper

This paper is organised as follows. Section 2 describes the documentation conventions I propose, including the pseudo-code. Section 3 uses them to specify and document a loans model, a simplified rational reconstruction of part of an Excel project-planner written by
one of my customers. This section shows various spreadsheets, stages in developing the model. They can be downloaded from the original version of this paper [Paine 2009a]. Finally, Section 4 is a short list of references.

2. DOCUMENTATION CONVENTIONS

2.1 Tables, table sizes, equations, and the purpose of cells

I want you to think in terms of "tables". By this, I don't mean pivot tables or anything else that Excel calls a table: I merely mean groups of logically related cells. Eventually, we'll map these into specific locations on worksheets, but when writing the documentation, I want you to regard as things in their own right.

When we think of a table, we must think of a name by which our documentation will refer to it. We must tell the reader how big it's going to be. We must also tell the reader what its purpose is, and make it clear exactly what it contains: how each cell relates to other cells, to the circumstances under which it is valid, and so on. THIS IS IMPORTANT, BUT OFTEN NEGLECTED IN DOCUMENTATION! And we must state the equations that define how to calculate the tables' elements.

To do these things, I shall use the following stylised documentation elements. Firstly, table definitions. These resemble the way one specifies functions in mathematics, and have the form:

```
table table name : bounds -> result type.
```

for example:

```
table total_cash_at_start_of_period : time_span -> currency.
```

The table name should clearly indicate the table's purpose. If the name contains more than one word, separate them by underlines, and choose the words to form a meaningful phrase.

The result type describes the kind of data each cell holds, and can be one of the cell-contents categories that Excel displays on the Number tab of its Format Cells dialogue: a name such as general, number, currency or date. We shall see later that I also use the name boolean, for cells that contain the logical values true or false.

The bounds specify the size of the table, and should be a name defined in the second kind of documentation element, a bounds definition. A bounds definition takes the form:

```
bounds bounds name: low bound to high bound.
```

For example:

```
bounds time_span: 1 to 12.
```

My reason for naming the bounds is that several tables may share the same bounds. In this cash-flow forecaster, for example, almost all the tables range over the same time span. Using a single name means you can define it in one place in the documentation, and easily update this when necessary.
The third kind of documentation element is an equation. This takes the form:

left-hand side = 
right-hand side.

For example:

\[ \text{total\_cash\_at\_end\_of\_period}[ t ] = \text{total\_cash\_at\_start\_of\_period}[ t ] - \text{expenses\_during\_period}[ t ] . \]

The right-hand side is an Excel formula, but with cell names replaced by table elements. Write table elements like array elements in a conventional programming language, with the table name followed by square brackets enclosing one or more indices.

The left-hand side is a table element where each index is: a constant; an "any value" variable; or an "any value" variable with a comparison that restricts its values. For example:

\[ \text{total\_cash\_at\_end\_of\_period}[ 1 ] \]
\[ \text{total\_cash\_at\_end\_of\_period}[ t ] \]
\[ \text{total\_cash\_at\_end\_of\_period}[ t > 1 ] \]

Finally, the fourth kind of documentation element is like a "comment" in a conventional programming language. It gives extra information about the purpose and content of a table, whenever the documentation elements above are not enough on their own. A comment consists of two dashes followed by English text, which will usually refer to one or more tables, table elements, or bounds. The dashes are to alert the reader quickly that it isn't one of the first three kinds of documentation element. For example:

-- total_cash_at_start_of_period[t] is the total cash held at the beginning of the first day of period t.

### 2.1 Documenting captions and other text

If you write documentation in this way, you will automatically end up with a description of almost every cell in your spreadsheet. How nice for those who will maintain or audit it!

The only cells you may not find yourself documenting are captions and other static text. Which is fine, because their purpose will normally be self-evident. But, if anything particularly unusual is required of these, you should document that too. For example, if certain labels need to be in merged cells, or in capitals, or intelligible to a user who doesn't know the jargon of your trade.

### 3. THE MODEL

I shall now develop a simple cash-flow and loans spreadsheet, a rational reconstruction of one used by one of my customers. Please note that I've simplified it in order to fit it into this paper, so it would need enhancement for real-world use. The loans, for example, don't have interest or monthly repayments. However, although simplified, it is based on a real-world application.
3.1 Modelling cash flow

Now I'll proceed with the spreadsheet, starting with the easiest part, the cash-flow tables. We shall forecast cash flow over monthly periods, displaying two tables. One shows total cash at the start of each period; one, total cash at the end of each period.

It's the recession, so let's assume there's no income, only expenses. Then total cash at the end of each period equals total cash at the start of each period minus expenses during the period. We can write this mathematically as:

\[
\text{total\_cash\_at\_end\_of\_period}(t) = \text{total\_cash\_at\_start\_of\_period}(t) + \text{expenses\_during\_period}(t).
\]

Here, I'm viewing \(\text{total\_cash\_at\_end\_of\_period}\), \(\text{total\_cash\_at\_start\_of\_period}\), and \(\text{expenses\_during\_period}\) as functions that map a period indicator \(t\) to money. It doesn't matter what \(t\) is as long as it uniquely identifies periods. For example, it could be an integer index, with 1 meaning 1st January 2009 to 31st January 2009, 2 meaning 1st February 2009 to 28th February 2009, and so on.

Now let's document this according to my principles. I shall make \(\text{total\_cash\_at\_end\_of\_period}\), \(\text{total\_cash\_at\_start\_of\_period}\), and \(\text{expenses\_during\_period}\) into tables. I need to add an \text{initial\_cash} table to give a starting value for \(\text{total\_cash\_at\_start\_of\_period}\). And I'm going to add a \text{time} table which shows the first date of each month. I shan't need this for my calculations, but will use it in captions.

I shall also need equations. One, for \(\text{total\_cash\_at\_end\_of\_period}\), will be very similar to the function above. But let's look at the whole lot:

bounds time\_span: 1 to 12.

table time : time\_span -> date.

-- time\([t]\) is the date of the first day of period \(t\). The final day of period \(t\) is the final day of its month, which is calculated below.

time\([t]\) =
    date( 2009, t, 1 ).

table initial\_cash : -> currency.

-- initial\_cash\([\]\) is the "opening cash balance", the value for the first period's \(\text{total\_cash\_at\_start\_of\_period}\). This will be input by the user.

table expenses\_during\_period : time\_span -> currency.

-- expenses\_during\_period\([t]\) is the expenses incurred during period \(t\): during the first day to the final day inclusive. This will be input by the user.
table total_cash_at_start_of_period : time_span -> currency.
    table total_cash_at_end_of_period : time_span -> currency.

-- total_cash_at_start_of_period[t] is the total cash
held at the beginning of the first day of period t.
Similarly, total_cash_at_end_of_period[t] is the total cash
held at the end of the final day of period t.

total_cash_at_start_of_period[ 1 ] =
    initial_cash[ ].

total_cash_at_start_of_period[ t>1 ] =
    total_cash_at_end_of_period[ t-1 ].

total_cash_at_end_of_period[ t ] =
    total_cash_at_start_of_period[ t ] - expenses_during_period[ t ].

Notice how careful I’ve been to say explicitly how the tables relate to the time periods. This is important! How many of us have coded half a cash-flow forecaster, only to discover an off-by-one error in our indexing, with the effect that — for example — cash for month m equals cash for month m minus expenses for month m. To which the only solution is that cash be either infinity or minus infinity: neither physically possible, even if the latter often feels as though it is.

Now, this is a very simple specification, but it’s still good to test it. So I’ve translated it into Excel. Tables, as I said are just groups of cells, and I’ve arranged these in columns, each headed by its name. This is a picture of it:

<table>
<thead>
<tr>
<th></th>
<th>A</th>
<th>B</th>
<th>C</th>
<th>D</th>
<th>E</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Time</td>
<td>Expenses during period</td>
<td>Initial cash</td>
<td>Total cash at</td>
<td>Total cash at</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>start of period</td>
<td>end of period</td>
</tr>
<tr>
<td>2</td>
<td></td>
<td></td>
<td>£100.00</td>
<td></td>
<td></td>
</tr>
<tr>
<td>3</td>
<td>01 January 2009</td>
<td>£5.00</td>
<td></td>
<td>£100.00</td>
<td>£95.00</td>
</tr>
<tr>
<td>4</td>
<td>01 February 2009</td>
<td>£5.00</td>
<td></td>
<td>£95.00</td>
<td>£90.00</td>
</tr>
<tr>
<td>5</td>
<td>01 March 2009</td>
<td>£5.00</td>
<td></td>
<td>£90.00</td>
<td>£85.00</td>
</tr>
<tr>
<td>6</td>
<td>01 April 2009</td>
<td>£5.00</td>
<td></td>
<td>£85.00</td>
<td>£80.00</td>
</tr>
<tr>
<td>7</td>
<td>01 May 2009</td>
<td>£5.00</td>
<td></td>
<td>£80.00</td>
<td>£75.00</td>
</tr>
<tr>
<td>8</td>
<td>01 June 2009</td>
<td>£5.00</td>
<td></td>
<td>£75.00</td>
<td>£70.00</td>
</tr>
<tr>
<td>9</td>
<td>01 July 2009</td>
<td>£5.00</td>
<td></td>
<td>£70.00</td>
<td>£65.00</td>
</tr>
<tr>
<td>10</td>
<td>01 August 2009</td>
<td>£5.00</td>
<td></td>
<td>£65.00</td>
<td>£60.00</td>
</tr>
<tr>
<td>11</td>
<td>01 September 2009</td>
<td>£5.00</td>
<td></td>
<td>£60.00</td>
<td>£55.00</td>
</tr>
<tr>
<td>12</td>
<td>01 October 2009</td>
<td>£5.00</td>
<td></td>
<td>£55.00</td>
<td>£50.00</td>
</tr>
<tr>
<td>13</td>
<td>01 November 2009</td>
<td>£5.00</td>
<td></td>
<td>£50.00</td>
<td>£45.00</td>
</tr>
<tr>
<td>14</td>
<td>01 December 2009</td>
<td>£5.00</td>
<td></td>
<td>£45.00</td>
<td>£40.00</td>
</tr>
<tr>
<td>15</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

In this picture, the money values are at most £100. That’s merely to narrow the cells so I can squeeze everything into this picture.

By the way, I tucked the "time" table out of the way on a sheet called Time. Whenever I want to use the dates in it as captions, I copy them. This separates how captions are calculated from how they are displayed. As Phil Bewig points out in his paper How do
you know your spreadsheet is right? Principles, Techniques and Practice of Spreadsheet Style [Bewig 2005], separating calculation from display is a Good Thing.

3.2 Adding loans to the program: the interface with cash flow

I'm now going to add loans to my spreadsheet. As before, I'll document this first, and only then translate the documentation into a spreadsheet. As a first step, I shall incorporate a borrowing term into the total_cash_at_end_of_period equation:

\[
\text{total\_cash\_at\_end\_of\_period}[t] = \\
\text{total\_cash\_at\_start\_of\_period}[t] - \\
\text{expenses\_during\_period}[t] + \\
\text{actually\_borrowed\_during\_period}[t].
\]

I'll also define two more tables. Into one of them, the user types the amount they want to borrow in any period. The other, actually_borrowed_during_period, will hold the amount the loans allow them to borrow. In this first-step version, I'll make the two equal:

\[
\begin{align*}
\text{table want\_to\_borrow\_during\_period} : \text{time\_span} & \to \text{currency}. \\
-- \text{want\_to\_borrow\_during\_period}[t] & \text{is the amount} \\
& \text{the user wants to borrow during period } t.
\end{align*}
\]

\[
\begin{align*}
\text{table actually\_borrowed\_during\_period} : \text{time\_span} & \to \text{currency}. \\
-- \text{actually\_borrowed\_during\_period}[t] & \text{is the amount} \\
& \text{the loans enable the user to borrow, taking} \\
& \text{borrowing limits into account. This will be less than} \\
& \text{or equal to want\_to\_borrow\_during\_period}[t].
\end{align*}
\]

\[
\text{actually\_borrowed\_during\_period}[t] = \\
\text{want\_to\_borrow\_during\_period}[t].
\]

Note that, once again, I have specified each table in pseudo-code, and also stated precisely what each cell in it means.

I'll now translate this into a spreadsheet. In the image below, you can see I've typed amounts of £20 and £10 into want_to_borrow_during_period. Please notice that these are mirrored in actually_borrowed_during_period:
3.3 Adding loans to the program: the loans themselves

Now, I'll add the loans stuff and change how actually_borrowed_during_period depends on want_to_borrow_during_period. Firstly, because I no longer want them to be equal, I remove that equation I added:

\[
\text{actually_borrowed_during_period}[t] = \text{want_to_borrow_during_period}[t].
\]

And secondly, I implement the loans. We shall have a fixed number of these. Each loan is like a credit card: you can borrow at any time, as long as the total you have borrowed doesn't exceed a ceiling. In the spreadsheet that inspired mine, there was also a "catch-all" loan with no ceiling: you could, in theory, borrow an infinite amount. I'll allow such loans too. So we need to define the number of loans, the ceilings of those that have ceilings, and a flag to say which do and which don't:

\[
\text{bounds loans_span: 1 to 4.}
\]

\[
\text{-- This gives the number of loans.}
\]

\[
\text{table has_ceiling : loans_span -> boolean.}
\]

\[
\text{has_ceiling}[l] \text{ is true if loan } l \text{ has a ceiling.}
\]

\[
\text{table ceiling : loans_span -> currency.}
\]

\[
\text{-- ceiling}[l] \text{ is } l\text{'s ceiling. That is, the user can borrow any amount } B \text{ such that } B \leq \text{ceiling}[l]. \text{ If not( has_ceiling}[l] ), the user can borrow without limit.} 
\]

<table>
<thead>
<tr>
<th></th>
<th>A</th>
<th>B</th>
<th>C</th>
<th>D</th>
<th>E</th>
<th>F</th>
<th>G</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>2</td>
<td></td>
<td>Time</td>
<td>Expenses</td>
<td>Initial cash</td>
<td>Total cash</td>
<td>Total cash</td>
<td>Want to</td>
</tr>
<tr>
<td></td>
<td></td>
<td>during period</td>
<td>during</td>
<td>£100.00</td>
<td>at start</td>
<td>at end of</td>
<td>borrow during</td>
</tr>
<tr>
<td>3</td>
<td>1/1/09</td>
<td>£5.00</td>
<td>£100.00</td>
<td>£95.00</td>
<td>£90.00</td>
<td>£0.00</td>
<td>period</td>
</tr>
<tr>
<td>4</td>
<td>1/2/09</td>
<td>£5.00</td>
<td>£95.00</td>
<td>£90.00</td>
<td>£85.00</td>
<td>£0.00</td>
<td></td>
</tr>
<tr>
<td>5</td>
<td>1/3/09</td>
<td>£5.00</td>
<td>£90.00</td>
<td>£85.00</td>
<td>£80.00</td>
<td>£0.00</td>
<td></td>
</tr>
<tr>
<td>6</td>
<td>1/4/09</td>
<td>£5.00</td>
<td>£85.00</td>
<td>£80.00</td>
<td>£75.00</td>
<td>£0.00</td>
<td></td>
</tr>
<tr>
<td>7</td>
<td>1/5/09</td>
<td>£5.00</td>
<td>£80.00</td>
<td>£75.00</td>
<td>£70.00</td>
<td>£0.00</td>
<td></td>
</tr>
<tr>
<td>8</td>
<td>1/6/09</td>
<td>£5.00</td>
<td>£75.00</td>
<td>£70.00</td>
<td>£65.00</td>
<td>£0.00</td>
<td></td>
</tr>
<tr>
<td>9</td>
<td>1/7/09</td>
<td>£5.00</td>
<td>£70.00</td>
<td>£65.00</td>
<td>£60.00</td>
<td>£0.00</td>
<td></td>
</tr>
<tr>
<td>10</td>
<td>1/8/09</td>
<td>£5.00</td>
<td>£65.00</td>
<td>£60.00</td>
<td>£55.00</td>
<td>£0.00</td>
<td></td>
</tr>
<tr>
<td>11</td>
<td>1/9/09</td>
<td>£5.00</td>
<td>£60.00</td>
<td>£55.00</td>
<td>£50.00</td>
<td>£0.00</td>
<td></td>
</tr>
<tr>
<td>12</td>
<td>1/10/09</td>
<td>£5.00</td>
<td>£55.00</td>
<td>£50.00</td>
<td>£45.00</td>
<td>£0.00</td>
<td></td>
</tr>
<tr>
<td>13</td>
<td>1/11/09</td>
<td>£5.00</td>
<td>£50.00</td>
<td>£45.00</td>
<td>£40.00</td>
<td>£0.00</td>
<td></td>
</tr>
<tr>
<td>14</td>
<td>1/12/09</td>
<td>£5.00</td>
<td>£45.00</td>
<td>£40.00</td>
<td>£35.00</td>
<td>£0.00</td>
<td></td>
</tr>
</tbody>
</table>

By the way, I reduced the dates to DD/MM/YY format. Again, that's to narrow the cells so they fit into this picture.
Do you see how I made has_ceiling distinct from the ceiling? I prefer this over a "nonsense value" convention such as that if a ceiling is -1 or #N/A, then it is infinite. Spreadsheeters often do rely on such nonsense values, but using a separate group of cells produces code that's easier to read and modify.

Now that I've described the parameters defining a loan, I shall model the loans' progress through time. First, I need an initial value, because the loans could have been arranged and borrowed from before the first cash-flow period. So I shall make a table analogous to initial_cash. Actually, you'll have seen it in the picture above:

```
table initial_loan : loans_span -> currency.
-- initial_loan[l] is the total already borrowed from loan l at time[1].
```

Now I define tables analogous to my total_cash tables. These record the progress of each loan over time:

```
table total_loan_at_start_of_period : loans_span time_span -> currency.
table total_loan_at_end_of_period : loans_span time_span -> currency.

-- total_loan_at_start_of_period[l,t] is the total lent by loan l at the start of period t, i.e. the total borrowed from that loan.
Similarly, total_loan_at_end_of_period[l,t] is the total lent by loan l at the end of period t.

total_loan_at_start_of_period[ l, 1 ] = initial_loan[ l ].
total_loan_at_start_of_period[ l, t>1 ] = total_loan_at_end_of_period[ l, t-1 ].
total_loan_at_end_of_period[ l, t ] = total_loan_at_start_of_period[ l, t ] + lent_during_period[ l, t ].
```

The equation for total_loan_at_end_of_period adds in a value for lent_during_period. This is the core of my spreadsheet, for it's here that we record which loan lends during each period, and how much. I'll get on to that later; for the moment, I'll just write up the table:

```
table lent_during_period : loans_span time_span -> currency.
-- lent_during_period[l,t] is the amount lent by loan l in period t.
```

Before explaining how lent_during_period works, I'm going to jump to another part of the calculation. This is where we decide which loans can be borrowed from at any time:
table can_supply_wants : loans_span time_span -> boolean.

-- can_supply_wants[l,t] is true if loan l is capable of lending what the user wants to borrow at period t. This does not imply that we will in fact use the loan.

can_supply_wants[l, t] =
    or( not( has_ceiling[l] )
    , want_to_borrow_during_period[t] +
    total_loan_at_start_of_period[l,t] <= ceiling[l] )
).

I like to read this as: can_supply_wants is a table of Boolean (logical) values. The element can_supply_wants[l,t] is true if l has no ceiling, or if the amount the user wants to borrow, plus the amount lent at the start of the period, is less than or equal to l's ceiling.

This table tells us which loans the user can borrow from, but how do we decide which they will in fact borrow from? To keep things simple, let's assume it's the first eligible loan, the one with the smallest l.

I am going to arrange can_supply_wants with time running vertically, and l running from left to right with no gaps. With this no-gaps layout, I can find the first true by scanning with Excel's match function:

table first_that_can_supply_wants : time_span -> general.

-- first_that_can_supply_wants[l,t] is the first l for which can_supply_wants[l,t] holds.
It is 1 if the user wants to borrow zero in period t (this is consistent), and #N/A if can_supply_wants[l,t] is false for all l, i.e. if no loan can supply what the user wants.

first_that_can_supply_wants[t ] =
    match( true, can_supply_wants[ all, t ], 0 )
).

Here is a picture of first_that_can_supply_wants, which I've laid out next to the cash tables. If you compare it with the ceilings above, you'll see that the first loan
with sufficient funds has been selected each time:

<table>
<thead>
<tr>
<th></th>
<th>A</th>
<th>B</th>
<th>C</th>
<th>D</th>
<th>E</th>
<th>F</th>
<th>G</th>
<th>H</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Time</td>
<td>Expenses</td>
<td>Initial</td>
<td>Total</td>
<td>Total</td>
<td>Want to</td>
<td>Actually</td>
<td>First that</td>
</tr>
<tr>
<td></td>
<td></td>
<td>during</td>
<td>cash</td>
<td>cash</td>
<td>cash</td>
<td>borrow</td>
<td>borrowed</td>
<td>can supply</td>
</tr>
<tr>
<td></td>
<td></td>
<td>period</td>
<td>start</td>
<td>at end</td>
<td>during</td>
<td>period</td>
<td>during</td>
<td>wants</td>
</tr>
<tr>
<td>2</td>
<td>1/09</td>
<td>£100.00</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>3</td>
<td>1/09</td>
<td>£5.00</td>
<td>£100.00</td>
<td>£110.00</td>
<td>£15.00</td>
<td>£15.00</td>
<td></td>
<td>1</td>
</tr>
<tr>
<td>4</td>
<td>1/09</td>
<td>£6.00</td>
<td>£110.00</td>
<td>£130.00</td>
<td>£25.00</td>
<td>£25.00</td>
<td></td>
<td>2</td>
</tr>
<tr>
<td>5</td>
<td>1/09</td>
<td>£5.00</td>
<td>£130.00</td>
<td>£170.00</td>
<td>£45.00</td>
<td>£45.00</td>
<td></td>
<td>3</td>
</tr>
<tr>
<td>6</td>
<td>1/09</td>
<td>£5.00</td>
<td>£170.00</td>
<td>£230.00</td>
<td>£65.00</td>
<td>£65.00</td>
<td></td>
<td>4</td>
</tr>
<tr>
<td>7</td>
<td>1/09</td>
<td>£5.00</td>
<td>£230.00</td>
<td>£225.00</td>
<td>£85.00</td>
<td>£0.00</td>
<td>£0.00</td>
<td>#/N/A</td>
</tr>
<tr>
<td>8</td>
<td>1/09</td>
<td>£5.00</td>
<td>£225.00</td>
<td>£232.00</td>
<td>£12.00</td>
<td>£12.00</td>
<td></td>
<td>2</td>
</tr>
<tr>
<td>9</td>
<td>1/09</td>
<td>£5.00</td>
<td>£232.00</td>
<td>£239.00</td>
<td>£12.00</td>
<td>£12.00</td>
<td></td>
<td>3</td>
</tr>
<tr>
<td>10</td>
<td>1/09</td>
<td>£3.00</td>
<td>£239.00</td>
<td>£246.00</td>
<td>£12.00</td>
<td>£12.00</td>
<td></td>
<td>4</td>
</tr>
<tr>
<td>11</td>
<td>1/09</td>
<td>£5.00</td>
<td>£246.00</td>
<td>£241.00</td>
<td>£12.00</td>
<td>£0.00</td>
<td></td>
<td>#/N/A</td>
</tr>
<tr>
<td>12</td>
<td>1/09</td>
<td>£5.00</td>
<td>£241.00</td>
<td>£236.00</td>
<td>£12.00</td>
<td>£0.00</td>
<td></td>
<td>#/N/A</td>
</tr>
<tr>
<td>13</td>
<td>1/09</td>
<td>£5.00</td>
<td>£236.00</td>
<td>£231.00</td>
<td>£12.00</td>
<td>£0.00</td>
<td></td>
<td>#/N/A</td>
</tr>
<tr>
<td>14</td>
<td>1/09</td>
<td>£5.00</td>
<td>£231.00</td>
<td>£236.00</td>
<td>£12.00</td>
<td>£0.00</td>
<td></td>
<td>#/N/A</td>
</tr>
</tbody>
</table>

And here is the table that first_that_can_supply_wants matches against, can_supply_wants, with its trues and falses. You can also see lent_during_period, showing the amount each loan has lent, and when:

<table>
<thead>
<tr>
<th></th>
<th>A</th>
<th>B</th>
<th>C</th>
<th>D</th>
<th>E</th>
<th>F</th>
<th>G</th>
<th>H</th>
<th>I</th>
<th>J</th>
</tr>
</thead>
<tbody>
<tr>
<td>25</td>
<td>1/1/09</td>
<td>TRUE</td>
<td>TRUE</td>
<td>TRUE</td>
<td>TRUE</td>
<td></td>
<td>£16.00</td>
<td>£0.00</td>
<td>£0.00</td>
<td>£0.00</td>
</tr>
<tr>
<td>26</td>
<td>1/2/09</td>
<td>FALSE</td>
<td>TRUE</td>
<td>TRUE</td>
<td>TRUE</td>
<td></td>
<td>£25.00</td>
<td>£0.00</td>
<td>£0.00</td>
<td>£0.00</td>
</tr>
<tr>
<td>27</td>
<td>1/3/09</td>
<td>FALSE</td>
<td>FALSE</td>
<td>TRUE</td>
<td>TRUE</td>
<td></td>
<td>£0.00</td>
<td>£0.00</td>
<td>£45.00</td>
<td>£0.00</td>
</tr>
<tr>
<td>28</td>
<td>1/4/09</td>
<td>FALSE</td>
<td>FALSE</td>
<td>FALSE</td>
<td>TRUE</td>
<td></td>
<td>£0.00</td>
<td>£0.00</td>
<td>£0.00</td>
<td>£65.00</td>
</tr>
<tr>
<td>29</td>
<td>1/5/09</td>
<td>FALSE</td>
<td>FALSE</td>
<td>FALSE</td>
<td>FALSE</td>
<td></td>
<td>£0.00</td>
<td>£0.00</td>
<td>£0.00</td>
<td>£0.00</td>
</tr>
<tr>
<td>30</td>
<td>1/6/09</td>
<td>TRUE</td>
<td>TRUE</td>
<td>TRUE</td>
<td>TRUE</td>
<td></td>
<td>£12.00</td>
<td>£0.00</td>
<td>£0.00</td>
<td>£0.00</td>
</tr>
<tr>
<td>31</td>
<td>1/7/09</td>
<td>FALSE</td>
<td>FALSE</td>
<td>TRUE</td>
<td>TRUE</td>
<td></td>
<td>£0.00</td>
<td>£0.00</td>
<td>£12.00</td>
<td>£0.00</td>
</tr>
<tr>
<td>32</td>
<td>1/8/09</td>
<td>FALSE</td>
<td>FALSE</td>
<td>FALSE</td>
<td>TRUE</td>
<td></td>
<td>£0.00</td>
<td>£0.00</td>
<td>£0.00</td>
<td>£12.00</td>
</tr>
<tr>
<td>33</td>
<td>1/9/09</td>
<td>FALSE</td>
<td>FALSE</td>
<td>FALSE</td>
<td>FALSE</td>
<td></td>
<td>£0.00</td>
<td>£0.00</td>
<td>£0.00</td>
<td>£0.00</td>
</tr>
<tr>
<td>34</td>
<td>1/10/09</td>
<td>FALSE</td>
<td>FALSE</td>
<td>FALSE</td>
<td>FALSE</td>
<td></td>
<td>£0.00</td>
<td>£0.00</td>
<td>£0.00</td>
<td>£0.00</td>
</tr>
<tr>
<td>35</td>
<td>1/11/09</td>
<td>TRUE</td>
<td>TRUE</td>
<td>TRUE</td>
<td>TRUE</td>
<td></td>
<td>£0.00</td>
<td>£0.00</td>
<td>£0.00</td>
<td>£0.00</td>
</tr>
<tr>
<td>36</td>
<td>1/12/09</td>
<td>TRUE</td>
<td>TRUE</td>
<td>TRUE</td>
<td>TRUE</td>
<td></td>
<td>£0.00</td>
<td>£0.00</td>
<td>£0.00</td>
<td>£0.00</td>
</tr>
<tr>
<td>37</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Let us now return to first_that_can_supply_wants and link it to lent_during_period. I said earlier that lent_during_period[l, t] is the amount lent by loan l in period t. Now I have everything necessary to calculate it:

```
lent_during_period[l, t] =
    if( isna( first_that_can_supply_wants[t] )
        , 0
        , if( l = first_that_can_supply_wants[t]
            , want_to_borrow_during_period[t]
            , 0
        )
    )
```

182
I shall read this as: the amount lent by loan l in period t is 0 if first_that_can_supply_wants\[t\] is #N/A: that is, if the match found no trues. Otherwise, if first_that_can_supply_wants\[t\] is l, it is the amount the user wants to borrow. Otherwise it is 0.

Ich am eldre and ek magti! At last, I can define actually_borrowed_during_period:

actually_borrowed_during_period[ t ] =
  sum( lent_during_period[ all, t ]).

This is just the sum over all loans of the amount lent during period t. In the pseudo-code, I've written the subscript as all to indicate that the sum ranges over the entire first dimension of lent_during_period.

And now, having finished the documentation, I build the spreadsheet by deciding where to put my tables, and then translating the equations into Excel formulae. (I know I showed pictures of it above, but that's because I went back over this posting and inserted them once I'd built it.) But the crucial point is that I now have a nice formal description of the spreadsheet, with every cell group documented.

4. References


EuSpRIG is Sponsored by:

This volume contains the proceedings of the tenth European conference:

‘The Role of Spreadsheets in Organisational Excellence’,
‘Le rôle des tableurs pour l'excellence dans les organisations’

EuSpRIG 2009 conference held in July 2009 at STEF ENS Cachan, Paris, France.

The objective of this conference is to promote discussion and co-operation 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.

ISBN          978-1-905617-89-0