18th EuSpRIG Annual Conference
“Spreadsheet Risk Management”

LSO KEY FINANCIAL DATA

YEAR

1993 1995 1997 1999 2001 2003 2005 2007 2009 2011 2013 2015

Thousands

FIXED
CASH
CAPITAL
INCOME

LSO FUTURE CASH

2016 2021 2026 2031 2036 2041 2046

Safety margin

Imperial Data Science Institute, Blackett Laboratory, Imperial College, London. Thursday 6th July 2017
EuSpRIG 2017 Conference

“Spreadsheet Risk Management”

EUSPRIG 2017 ORGANISING COMMITTEE

Patrick O’Beirne (Chair), Systems Modelling, Co. Wexford, Ireland
David Colver (Treasurer), Operis, London, UK
Grenville J. Croll, Bury St Edmunds, UK
Simon Thorne, Cardiff Metropolitan University, UK
Angela Collins, (Secretary) Manchester, UK

EUSPRIG 2017 PROGRAMME COMMITTEE

Simon Thorne (Chair), Cardiff Metropolitan University, UK
Grenville J. Croll, Spreadsheet Engineering, Bury St Edmunds, UK
Patrick O’Beirne, Systems Modelling, Co. Wexford, Ireland
EUSPRIG 2017 CONFERENCE PROGRAMME

Thursday 6th July 2017, Imperial Data Science Institute, Blackett Laboratory, Imperial College

08:30 – 09:00  Arrival, Registration, Coffee and welcome

09:00 – 10.30  Session 1

The Future of Spreadsheets in the Big Data Era, Imperial College, David Birch, David Lyford-Smith & Yike Guo

Alternative Spreadsheet Model Designs for an Operations Management Model Embedded in a Periodic Business Process, University of San Francisco, Thomas A. Grossman, Vijay Mehrotra, Mouwafac Sidaoui

10:30 – 11:00  Session 2

Structured Spreadsheet Modelling and Implementation with multiple dimensions part 1 modelling, Paul Mireault, SMMI International

11:00 – 11:30  Coffee Break

11:30 – 12:30  Session 3

Mitigating Spreadsheet Risk in Complex Multi-Dimensional Models in Excel
Steve Litt, Indigo Sun, Incorporated

Proposed Spreadsheet Transparency definition and measures, Craig Hatmaker

12:30 – 14:00  Lunch and visits to the Data Observatory

14:00 – 15.30  Session 4

The Reification of an Incorrect and Inappropriate Spreadsheet Model, Grenville J. Croll

Electronic Evidence and the Presumption of Reliability, Stephen Mason

Mitigating Spreadsheet Model Risk with Python Open Source Infrastructure
Oliver Beavers, Trivium Financial Group

15:30 – 16.00  Tea break

16:00 – 17:30  Session 5

Structuring Spreadsheets with the “Lish” Data Model, Alan Hall, Michel Wermelinger, Tony Hirst and Santi Phithakkitnukoon.

Edu-Edition Spreadsheet Competency Framework, Maria Csernoch, Piroska Biró, Univ. Debrecen


17.30  Conference closes. Food and drink for the evening will be found locally. Please come along and enjoy the post conference discussions. Details to be confirmed
# EUSPRIG 2017 PROCEEDINGS

## TABLE OF CONTENTS

<table>
<thead>
<tr>
<th>Author(s)</th>
<th>Paper</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>David Birch, David Lyford-Smith, Yike Guo; Imperial College</td>
<td>The Future of Spreadsheets in the Big Data Era</td>
<td>1</td>
</tr>
<tr>
<td>Thomas A. Grossman, Vijay Mehrotra, Mouwafac Sidaoui; University of San Francisco</td>
<td>Alternative Spreadsheet Model Designs for an Operations Management Model Embedded in a Periodic Business Process</td>
<td>15</td>
</tr>
<tr>
<td>Stephen Mason</td>
<td>Electronic Evidence and the Presumption of Reliability</td>
<td>27</td>
</tr>
<tr>
<td>Steve Litt, Indigo Sun Inc.</td>
<td>Mitigating Spreadsheet Risk in Complex Multi-Dimensional Models in Excel</td>
<td>35</td>
</tr>
<tr>
<td>Craig Hatmaker</td>
<td>Proposed Spreadsheet Transparency definition and measures</td>
<td>49</td>
</tr>
<tr>
<td>Grenville J. Croll EuSpRIG</td>
<td>The Reification of an Incorrect and Inappropriate Spreadsheet Model</td>
<td>63</td>
</tr>
<tr>
<td>Paul Mireault SMMI International</td>
<td>Structured Spreadsheet Modelling and Implementation with multiple dimensions. Part 1: modelling</td>
<td>77</td>
</tr>
<tr>
<td>Oliver Beavers Trivium Financial Group</td>
<td>Mitigating Spreadsheet Model Risk with Python Open Source Infrastructure</td>
<td>91</td>
</tr>
<tr>
<td>Alan Hall, Michel Wermelinger, Tony Hirst, Santi Phithakkitnukoon.</td>
<td>Structuring Spreadsheets with the “Lish” Data Model</td>
<td>107</td>
</tr>
<tr>
<td>Maria Csernoch, Piroska Biró University of Debrecen</td>
<td>Edu-Edition Spreadsheet Competency Framework</td>
<td>121</td>
</tr>
<tr>
<td>Simon Thorne, Cardiff Metropolitan University, The Medical Algorithms Company</td>
<td>The Role of Spreadsheets in Clinical Decision Support: A Survey of the Medical Algorithms User Community</td>
<td>137</td>
</tr>
</tbody>
</table>
PREFACE

Dear Colleagues,

You are very welcome to the Eighteenth Annual Conference of the European Spreadsheet Risks Interest Group. Currently we have about 6,000 unique visitors to our website per month, the most popular page being the horror stories.

The papers this year follow three main threads, which go way beyond the simple prescriptions of the past and require a distinct sharpening of skills in data structuring.

Five papers, the majority of the conference, tackle ways to deal with real world data which does not fit easily into the Procrustean bed of the two-dimensional grid. They are: Structuring Spreadsheets with the “Lish” Data Model, Alternative Spreadsheet Model Designs for an Operations Management Model, Mitigating Spreadsheet Risk in Complex Multi-Dimensional Models, Structured Spreadsheet Modelling and Implementation with Multiple Dimensions, and The Future of Spreadsheets in the Big Data Era. Three papers deal with the issues of the reliability of the data and the suitability of using a spreadsheet at all: Electronic evidence and the presumption of reliability, The Role of Spreadsheets in Clinical Decision Support, and The Reification of an Incorrect and Inappropriate Spreadsheet Model. Three more deal with the more familiar topics of readability, skill sets, and testing: Proposed Spreadsheet Transparency Definition and Measures, Edu-Edition Spreadsheet Competency Framework, and Mitigating Spreadsheet Model Risk with Python.

This will be a packed schedule dealing with some topics that will demand your attention and stimulate debate that will go on both in informal meetings afterwards, and in our online discussion forum on Yahoo groups which has over 900 members.

I would like to acknowledge with gratitude our sponsors who have supported us financially and professionally. Our logo sponsors this year are CIMCON, and I recommend you to pick up their brochures and learn about their services. And our venue sponsor is Imperial College London, for the use of whose facilities we are very grateful.

It is my pleasure to once again acknowledge the keen work of our conference and programme organiser Simon Thorne from Cardiff Met. The committee also depends upon the wise counsel and active support of David Colver of Operis, Grenville Croll of Spreadsheet Engineering, Angela Collins (secretary) and David Birch (Imperial College London). Representatives of these organisations have contributed a great amount of expertise in the organising of this conference, the publicity, the proceedings, and much more committee work in the background.

Thank you for your interest and participation, and we look forward to a stimulating and interactive conference!

Patrick O'Beirne, Chair 2017
The Future of Spreadsheets in the Big Data Era

David Birch1*, David Lyford-Smith2 & Yike Guo1
1Data Science Institute, Imperial College London
2ICAEW IT Faculty
*david.birch@imperial.ac.uk

ABSTRACT

The humble spreadsheet is the most widely used data storage, manipulation and modelling tool. Its ubiquity over the past 30 years has seen its successful application in every area of life. Surprisingly, the spreadsheet has remained fundamentally unchanged over the past three decades. As spreadsheet technology enters its 4th decade a number of drivers of change are beginning to impact upon the spreadsheet. The rise of Big Data, increased end-user computing and mobile computing will undoubtedly increasingly shape the evolution and use of spreadsheet technology.

To explore the future of spreadsheet technology a workshop was convened with the aim of “bringing together academia and industry to examine the future direction of spreadsheet technology and the consequences for users”. This paper records the views of the participants on the reasons for the success of the spreadsheet, the trends driving change and the likely directions of change for the spreadsheet. We then set out key directions for further research in the evolution and use of spreadsheets. Finally, we look at the implications of these trends for the end users who after all are the reason for the remarkable success of the spreadsheet.

1. INTRODUCTION

The goal of this paper is to explore the future of the spreadsheet in the light of the age of Big Data. Firstly, we explore the reasons underlying the success of spreadsheet technologies over the past 3 decades before exploring the challenges encountered using current spreadsheet technologies, many of which such as high error rates are well-known, at least within the research community.

As computing continues to evolve, we then explore the “Drivers of Change” which the authors believe are likely to impact upon the development and use of the future spreadsheet. Key amongst these drivers is the rise of Big Data and Artificial Intelligence (AI) which will be explored in depth. These drivers are likely to result in particular “Directions of Change” specific to the evolution of spreadsheets. Key amongst these directions of change include better support for dealing with large external datasets and the ability to integrate such data together.

How spreadsheet technology should adapt in response to drivers of change presents a range of open research questions, we outline the most compelling research questions in section 6. Finally, we conclude this paper with a look at the implications for spreadsheet users who after all have made the spreadsheet such a success and will drive spreadsheets success far into the future.

2. THE SUCCESS OF THE SPREADSHEET

Before commenting on the future of the spreadsheet it is necessary to explore the factors underlying the spreadsheets enduring success. From literature, discussions during the workshop and the author’s experiences, we identify the following factors:

a) Ubiquity

Spreadsheets are universally available to students and professionals, indeed most computer literate people have a familiarity with spreadsheets. Spreadsheets exist on most operating systems, architectures and form factors, this has made them a common tool and format for working with and exchanging data, encoding calculations and modelling business situations.
b) Unconstrained canvas

A blank spreadsheet is just that: blank. It may become a shopping list, a financial model or a 250,000 cell multi-disciplinary model. This flexibility, while it is the spreadsheet’s biggest asset, frequently, proves it’s Achilles’ heel, as we will highlight. This adaptability has led to spreadsheets being applied in nearly every industry and situation. One of the key reasons for this is the openness of the spreadsheet and the clarity of its mental model – all a user needs to understand is the concept of tables of cells which may contain text, numbers or dates or may be formulas calculating with such data.

c) An open box

Unlike many business applications, spreadsheets are inherently open, there is no “black box”; everything is open and can, in theory, be understood and changed cell by cell. This, of course, does not always happen in practice as user’s self-confidence and the sheer complexity of large formulas and models may overwhelm and are known to cause hidden errors in spreadsheets [Panko 2015].

d) Ability to communicate data

Spreadsheets provide simple-to-use data visualisation tools which enable communication of quantitative insight succinctly to a wide audience. They also provide an early way of understanding and exploring data to discover trends and patterns. Since these visualisations are often based on the results of calculated data, they turn into an interactive graphical tool for visually answering questions about a modelled case study. As spreadsheets are a standard format these interactive models or summaries of data can be shared visually to a wide audience.

e) Ability to store data in a computable format

For many end users the primary use of a spreadsheet is to store data. A spreadsheet’s expandable tabular format enables extensible storage of data. Key to their utility is this computability of this format, from a simple sum of a table to a depreciation calculation to a mail merge, making data computable is a key feature behind the success of spreadsheets.

f) Lightweight databases

Coupled with the ability to store data in a computable format we note that the spreadsheets often act as a lightweight database. This is particularly apparent for users without training in or access to more advanced database systems. Recently analysis of a large companies spreadsheets showed that one third of files contained only data and no formulas [Hermans 2015]. Spreadsheets allow the storage of data in a format which allows the look up of data items, the filtering of data to meet criteria and the “joining” of data tables which are the basic features of a database. This feature trend is increasing with systems such as Microsoft’s PowerPivot providing tools to summarise data tables graphically data directly as if the user were writing in a query language such as SQL.

g) Ability to model and test scenarios

Another common use of spreadsheets is to act as a computable “back of the envelope” for carrying out calculations to model scenarios. Once created these models are then able to be used to test a variety of scenarios with the same calculation logic. This enables users to gain an understanding the possible design space of outcomes. The spreadsheet’s ability to encode complex logical, statistical or engineering calculations assists users to carry out tasks which would be challenging manually or would otherwise require customised software or the ability to program.

h) Accessible End-User Programming

Spreadsheets are widely regarded as the most common means of end user programming [Scaffidi 2005]. The simple mental model and “what you see is what you get” environment has enabled many millions of people to create complex calculation algorithms implemented naturally and incrementally. This has enabled spreadsheets to be adapted to many specific scenarios.

i) Advanced Functionality

A key feature enabling the spreadsheet’s success is its ability to grow with the end-users competence. Initially the model of spreadsheet appears simple it may be augmented by numerous complex formulas and algorithms. From experimenting with logical and lookup functions through the creation of Macros to the writing of complex scripts in language such as Visual Basic for Applications, most spreadsheet systems provide advanced functionality to enable users to accomplish complex tasks.
j) Integrated Development Environment
As identified by [Grossman 2007] the spreadsheet provides an integrated development environment which supports end users in their development of business critical applications. Support provided in terms of immediate feedback, error correction and debugging tools enables rapid prototyping and experimentation in the design and modification of spreadsheet models. It is key that development occurs within an already familiar environment for which users have a well-known mental model and which can be shared with other users.

k) Ability to formalise a process
Finally in the business context spreadsheets are often formalising a business process [Buckner 2004], for example submitting an expenses claim. Such models can be tailored specifically for the use case and importantly this can be done by the people who will be using the model who are empowered to evolve the model as needs change without the need for complex software projects.

Conclusions:
The key strengths of the spreadsheet stem from its simple model of tables of cells containing data or formulas. This model has been grasped by a wide audience and used for a wide variety of purposes. Underlying this is the model’s adaptability and flexibility, providing few constraints to usage in a common sharable format. Supporting this simplicity is the provision of a wide set of advanced features supporting more specialised tasks, these are provided on a continuum hiding advanced features from novice users presenting a simple interface. Finally the interoperability of these features combined with a wide user base has enabled the global success of the spreadsheet.

3. CHALLENGES WITH CURRENT SPREADSHEET TECHNOLOGY

While these features of spreadsheets have led its success, they have also caused a number of challenges. Some are known to daily spreadsheet users, while others are only widely known within professional and research communities. Addressing these challenges has been the subject of research for at least the past decade. Here we outline key challenges with spreadsheet technologies:

a) Hidden Errors
Humans are fallible creatures, hence it is widely known in literature [Panko 1998] [Powell 2008] [Panko 2015] (though perhaps not in wider practice) that the error rate in the creation, evolution and use of spreadsheets is strikingly high. This is known to be the case not only in complex formulas but in data entry and use of many spreadsheet features. Overconfidence among users coupled with a lack of testing and the hidden nature of the structure of how formulas interact combine to result in a recurrent stream of horror stories in finance, industry [Thorne 2013] and research [Herndon 2014].

b) Comprehensibility
One underlying reason for this high error rate is known to be the user’s difficulty in understanding the spreadsheet they are interacting with [Panko 2015] [Kohlhase et al 2015]. Formulas are hidden from view by default making understanding of logic, calculation and the flow of data through the model a challenge. Further it is a challenge to understand the wider hidden structure of the model, particularly when formulas interact across worksheets out of the immediate view of users. Such challenges mean that gaining a high level view of how a spreadsheet model works remains difficult [Hermans 2011].

c) Complexity
Spreadsheets have a tendency to grow over time, both in the amount of data stored and the complexity of the calculations performed (for an extreme example see [Birch et al 2014]). Without clear guidelines and a clear purpose the complexity of a spreadsheet is unbounded and challenging to deal with, particularly when such models are inherited by new users. There are few mechanisms to compartmentalise complexity within spreadsheets, such as the role functions, classes and modules provide within more general programming.

d) Adaptability and informality
One of the great strengths of spreadsheets is their adaptability however this causes a great weakness, spreadsheets often evolve to become unmanageable and error prone. With no rules or guidance for development of a model its complexity will tend increase and its integrity to degrade as the model is
reused and adapted to new situations. This is particularly noticeable when a model is shared or inherited between multiple users. To address this a number of spreadsheet development guides [Grossman 2010] and processes [Ferreira and Visser 2012] have been developed though not widely adopted outside of their context.

e) Limited scale
Current spreadsheet technologies are unable to scale to support large datasets which are increasingly encountered. While the maximum number of rows Microsoft Excel supports has increased from $65536 (2^{16})$ to $1,048,576 (2^{20})$ over the last decade, this limit is still frequently less than the data encountered and which professionals seek to gain insight from.

f) Limited formal training
A large proportion of spreadsheet users are, beyond the basics, primarily self-taught, meaning that many will have picked up bad habits through lack of knowledge, or will be unaware of techniques that could help reduce their error rate or increase their efficiency.

Conclusions:
Comprehending the complexity of a spreadsheet model to generate a good mental model of the structure of the spreadsheet remains the key challenge for professionals working with spreadsheets. Doing so while identifying and avoiding hidden errors in spreadsheets requires good training and the disciplined use of the adaptability and informality which is one the greatest strengths of the spreadsheet model. Augmenting this integrated development environment to assist management of this complexity remains a key challenge which will only grow as we approach scale limits of data supported by modern spreadsheets. This limitation is symptomatic of the one of the key Drivers of Change facing the spreadsheet – the rise of “Big Data”.

4. DRIVERS OF CHANGE

The evolution of technology has and will continue to impact upon the evolution of the spreadsheet and affect their users. These drivers of change may impact gradually or bring rapid, fundamental change. It remains to be seen whether this will require augmentation of the spreadsheet as we know it with new features and tools or whether a more fundamental change of the underlying spreadsheet model will be required.

a) The rise of Big Data
It is now increasingly reasonable to collect and collate large corpuses of data which far exceed the ability of spreadsheets to store, manipulate or analyse. While common databases and specialist tools do exist for interacting with such data, these tools are beyond the experience or skillset of most spreadsheet users. This will decrease the effectiveness and the utility of spreadsheets in general practice, particularly as the opportunities and insight within large datasets are becoming increasingly valuable even for small companies. Analysing patterns in hundreds of thousands of historical orders or analysing millions of website traffic interactions are tasks increasingly common, even in small firms who will not have familiarity, expertise or access to more specialised tools than the humble spreadsheet.

b) The heterogeneity of data
As more data is collected, the variety of the data will correspondingly increase. There will be an increasing need to relate, fuse and analyse multiple datasets concurrently. This is frequently beyond the capability of spreadsheets and their lookup functions requiring tools traditionally only found in databases. Data will also no longer take traditional forms with new data including unstructured textual and image-based data. This data is beyond the scope of current spreadsheet technology.

c) Coping with unreliable data
Inevitably as the quantity of data to be processed grows additional challenges with its quality will be encountered. Current spreadsheets struggle to cope with malformed or inconsistent data and require many manual steps to reformate and coerce data into a clean state for working with. This assessment of the veracity of the underlying data is a key step for modern data science and will become increasingly important yet is hard to perform with existing spreadsheets.
The Future of Spreadsheets in the Big Data Era

The Success of the Spreadsheet

Ubiquity
Unconstrained canvas
An open box
Ability to communicate data
Ability to store data in a computable format
Lightweight databases
Ability to model and test scenarios
Accessible End-User Programming
Advanced Functionality
Integrated Development Environment
Ability to formalise a process

Drivers of Change

The rise of Big Data
The heterogeneity of data
Coping with unreliable data
The rise of machine learning
The rise of Artificial Intelligence
Rise of real-time data and the Internet of Things (IOT)
The rise of cloud computing and Software as a Service
The rise of the App ecosystem
Online collaboration suites
Increased regulatory understanding of risk
Enterprise spreadsheet management
A more computer literate generation
A change of end user computing devices
The rise of Business Intelligence software
The rise of data dashboard systems

Directions of Change

Analysis of external Big Data
Heterogeneous unreliable data
Data fusion
Data analysis tools
Exploratory data visualisation
Cloud based analysis
Improved Auditing tools

Research Directions

Supporting End user Programming
Structuring of spreadsheets
Identification and auditing of errors
Refactoring tools for spreadsheets
Mechanisms for Code Reuse
Open source model repositories
Dealing with Data
Data analysis techniques
Version control and collaboration in spreadsheets
Interacting with spreadsheets in new form factors

Challenges with Spreadsheets

Hidden Errors
Comprehensibility
Complexity
Adaptability and informality
Limited scale
Limited formal training

Lightweight databases
Ability to store data in a computable format
Ability to model and test scenarios
Accessible End-User Programming
Advanced Functionality
Integrated Development Environment
Ability to formalise a process

Analysis of external Big Data
Heterogeneous unreliable data
Data fusion
Data analysis tools
Exploratory data visualisation
Cloud based analysis
Improved Auditing tools

Supporting End user Programming
Structuring of spreadsheets
Identification and auditing of errors
Refactoring tools for spreadsheets
Mechanisms for Code Reuse
Open source model repositories
Dealing with Data
Data analysis techniques
Version control and collaboration in spreadsheets
Interacting with spreadsheets in new form factors

Limited scale
Limited formal training
d) The rise of machine learning
Supervised and unsupervised machine learning from data either in classification of data or the prediction of trends from historical data are becoming key techniques for the modern data scientist. Such methods are not found within modern spreadsheets, partly because they are unable to cope with the scale of data required for successful learning which will be an increasingly key technique.

e) The rise of Artificial Intelligence
The availability of larger volumes of data, computing power and machine learning techniques such as deep learning have empowered the rise of Artificial Intelligence which will impact upon the workplace in the coming years. Reports by [Frey & Osborne 2013] have suggested that many current administration and data processing jobs will be supported or replaced by AI systems in the future. Examples include the rise of digital assistances such as Amazon’s Alexa, Microsoft’s Cortana and Apple’s Siri. Such technology also appears in online shopping assistants or chat-bots. One example of the impact this may have is the use of an AI digital assistant to record the annual leave taken in an office, a role traditionally fulfilled by administration staff using a spreadsheet. This may greatly reduce the requirement for spreadsheet use. It may also be that digital assistants will help with the design of new spreadsheet models.

f) Rise of real-time data and the Internet of Things (IOT)
As surveyed by [Atzoria 2010] the Internet of Things is the convergence of several enabling hardware and communications technologies which enables the instrumentation of everyday situations and objects. The flow of information this provides typically exceeds the scale of spreadsheets to analyse. Such analysis is often time critical requiring the generation of realtime alerts, something beyond existing spreadsheets.

g) The rise of cloud computing and Software as a Service
Cloud computing has offered the opportunity to open traditionally closed business systems to the world online. Such systems are termed “Software as a Service” [Turner 2003] and present complex functionality (for example small business accounting) through simple websites with complex processing of data carried out within the cloud. This limits the need for the development and management of software and IT infrastructure. Many of the business roles which is now offered as “Software as a Service” would traditionally have been fulfilled by spreadsheets managed by professionals. While some systems enable using spreadsheets as the calculation engine of software as a service (e.g. Google Forms) this presents a challenge to the existing role of spreadsheets.

h) The rise of the App ecosystem
Many spreadsheets are created to fulfil specific purposes where no other software exists. The rise of the App eco-system on many platforms provides a wide range of niche applications. The challenge potentially facing spreadsheets is “Why create a spreadsheet model if “there’s an app for that”?”.

i) Online collaboration suites
The rise of online office suites such as Google Docs and Office 365 is changing the nature of the way we use spreadsheets. These systems enable real time collaborative updates to a single document, they also enable the use of crowd sourcing of data and insight from a large number of participants. This is a new way of working with spreadsheets with multiple authors concurrently editing a document, this leads to more efficient collaboration and sharing of documents. However it also leads to challenges around ownership and the integrity of constantly evolving models which makes tracing the provenance of calculated information a challenge. Underling this is a version control, a widely known tool in the software development community but a newer challenge to spreadsheets.

j) Increased regulatory understanding of risk
Over the past 15 years a number of accounting scandals [Croll 2009] have been linked to errors in spreadsheet models. This combined with the Financial Crisis of 2008 has led to an increased focus on controlling risk within financial models, many of which are built as spreadsheets. This has led to a demand for auditing tools for spreadsheets in an attempt to alleviate the level of risk.

k) Enterprise spreadsheet management
Given this increased awareness of risk many enterprises are keen to take “control” over spreadsheet development and usage. This may take the form of an auditing team or specific training in spreadsheet...
development. Alternatively spreadsheets may be used to create the initial model or algorithm before being transformed by a development team into an authorised line of business application. Adapting spreadsheets to support this transition smoothly will be key to their utility in the future.

1) A more computer literate generation
Driven by the rise of code academies and the re-introduction of programming into the school curriculum in the UK the next generation of professionals will increasingly seek tools which enable them to fluently code bespoke solutions to their problems. This will increase use of advanced features of spreadsheets (for example macros and scripting) and simultaneously an abandonment of spreadsheets which are found to be “too simple” or “too constraining” for literate programmers.

m) A change of end user computing devices
The rise of mobile computing on tablets and smart phones and its new form-factor is now beginning to affect the workplace. Some office applications have been adapted to mobile phones with varying degrees of success as devices are generally optimised for consuming content rather than creating it. Spreadsheets are difficult to work with on such devices and this may affect their future usability.

n) The rise of Business Intelligence software
Such programs such as [Tableau, Spotfire, PowerBI] offer users a visual-first approach for exploring and analysing data by enabling large datasets to be graphed across many different data sources including database servers. Such systems are often used to create data dashboards.

o) The rise of data dashboard systems
One role traditionally performed by spreadsheets is the regular presentation of data as a dashboard of performance. This role, once the preserve of spreadsheets, is increasingly now filled by specialised systems, often implemented in a Software as a Service model which allow the creation of graphical dashboards updated in real-time by querying database servers. Providing online or intranet dashboards these systems can be interrogated by a wider audience. While Microsoft’s Excel has begun to move in this direction via the integration of dashboards in PowerBI the challenge to spreadsheets will increase.

Conclusions:
These drivers present a threat to the dominance of the spreadsheet as a key tool for working with data and for modelling by future professionals. These threats take three forms:

1. The rise of increasingly available Big Data, its scale and variety combined with its unreliability mean that spreadsheets are no longer able to support it. Such data enables insightful methods of machine learning which are not supported by spreadsheets and which will power AI systems which may replace spreadsheets in some contexts.
2. A change in how spreadsheets are used, resulting in new demands. As seen in the rise of online collaboration and a change in the type and form factor of devices we use daily.
3. Specialised competitors to the spreadsheets including Business Intelligence suites for analysing and presenting data or on a wider scale the rise of App stores and Software as a Service systems offering fully featured software in cases where spreadsheets may once have been created. This is supported by an increased understanding of the risk of spreadsheet models and a desire for enterprise control over them.

These drivers of change present exciting opportunities to empower professionals with new tools and techniques to master the challenges of computing with data in the next decade. Alternatively new, more specialised, tools will be developed and mastered by (new?) professionals, this will lead to a side-lining of the venerable spreadsheet.

5. DIRECTIONS OF CHANGE
Given the drivers of change identified we now explore the likely directions of change in which spreadsheets are likely to evolve. It is unclear whether such evolution can naturally retain the underlying simplicity of the spreadsheet model or whether a revolution in the concept of a spreadsheet will be required. The authors believe that in the future spreadsheets will support:


a) **Analysis of external Big Data**

As the volume of data increases spreadsheets will be forced to provide mechanisms for working with such Big Data. Given the scale of this data it will be necessary for it to be stored in external databases or files or APIs. Such data can then be queried, ingested, transformed and analysed through the spreadsheet engine in much the same way that large tables within spreadsheets are today. This trend is already in evidence with Microsoft’s PowerBI tool suite [PowerBI]. This may also be powered by the idea of taking your algorithm to the data (residing in the cloud) rather than bringing the data to your calculation spreadsheet. This will require the ability to extract algorithms from spreadsheets.

b) **Heterogeneous unreliable data**

Spreadsheets will increasingly be called upon to analyse data types beyond structured numeric data. For example textual and unstructured document data, much of which is unreliable and has missing data points. As the volume of this data grows tools will be needed to deal with the transformation of such data with a better range of error states for the future spreadsheet.

c) **Data fusion**

While existing lookup functions provide the ability to relate datasets together within the spreadsheet there will be an increasing need to cross-reference and join external datasets together. This is one of the traditional strengths of SQL databases, though these tools are often too complex for most end users. Tools such as [Tableau], [Spotfire] and [PowerBI] are beginning to support this however native support within a spreadsheet would empower a wider audience.

d) **Data analysis tools**

Future spreadsheets must incorporate the new generation of data analysis tools, ranging from machine learning algorithms for classification of data or prediction to more advanced statistical tools for model testing. These tools need not only be applied in their traditional role but may be used to directly assist the spreadsheet user, one example of this trend is Excel’s Flash Fill function which predicts additional or missing data items in a series.

e) **Exploratory data visualisation**

Traditionally data visualisation is used to communicate the results of analysis, a newer trend is to use interactive visualisation to explore large datasets identifying insight visually. This is evidenced by the success of tools such as [Tableau], [Spotfire] and [PowerBI]. Although these tools are excellent for exploring data they lack the flexibility for working and calculating with data which spreadsheets traditionally provide.

f) **Cloud based analysis**

The future of the spreadsheet will increasingly be affected by online collaboration, with concurrent editing of spreadsheets stored in a single authoritative repository. This presents the opportunity for the application of large amounts of computing to be applied to models for additional insight through techniques such as sensitivity analysis [Birch et al 2014]. Cloud based hosting will also enable models to be published to form the basis of an online application or service.

g) **Improved Auditing tools**

Given the increasing recognition of the danger and prevalence [Panko 2015] of spreadsheet errors combined with their hidden nature it is increasingly necessary for auditing tools to be developed and applied to spreadsheets. These may take the form of guidelines for modelling [Grossman 2010], or automated tools for detecting suspicious calculation formulas [Hermans 2012]. Further tools to support users with gaining an understanding of the structure of large models through techniques such as visualisation [Hermans 2011] should be employed. Underlying all of these techniques are an adopting of standard techniques and tools for supporting programmers to the world of spreadsheets. It is very likely that this trend will continue as more techniques are usefully adopted into the end-user programming domain.

**Conclusions:**

In light of the drivers of change identified the authors believe that the central direction of change in the future of the spreadsheet will be to aid spreadsheets in continuing to fulfil their daily role – dealing with data. As data grows in scale, becomes more heterogeneous and is less reliable better
tools and support will need to be integrated into the spreadsheet model. The aggregation of cells into Table structures within a spreadsheet is the start of this trend, spreadsheets must continue to be expanded to work fluently with such Big Data, and perhaps other useful spreadsheet structures will emerge, for example “Table Summary”, “Table Join”, “Database Query”. Working with such data will drive a need to join or fuse datasets together, something which is traditionally tricky in spreadsheets via lookup functions and is the preserve of databases. Once spreadsheets are able to cope with such data tools will be needed to analyse and explore it. Machine learning for prediction of trends or classification of data is one technique well suited for bigger datasets which will need to be integrated into spreadsheets. Interactive visualisation of data for exploration is an increasingly common requirement and one that spreadsheets are beginning to adapt to [PowerBI]. The central challenge for the future will be how to adapt the spreadsheet model toward these directions without damaging the central successful paradigm of spreadsheets that make them so valuable to end users.

6. RESEARCH DIRECTIONS

Taking advantage of the opportunities presented for spreadsheets opens a number of fruitful research areas, among those identified we believe the most important are:

   a) Supporting End user Programming
For most end users programming is a challenging prospect, spreadsheets provide a familiar context for understanding common concepts in programming and extending their knowledge. Supporting end users comprehension of spreadsheet models and programming concepts within is an active area of research [Hermans 2011]. Three key directions of research which assist in this are as follows:

   b) Structuring of spreadsheets
Most spreadsheets have an internal structure containing tables, calculation areas and summaries, however few of these are treated as high level objects such as tables. Adding further high level structures to spreadsheets would improve the comprehensibility of models these might include external data imports, clusters of cells performing a calculation (the equivalent of a method in a programming language). One key missing component is the equivalent of a program class – a representation of an entity containing a group of properties about that object. Such entities may be discoverable within spreadsheet models [Abraham and Erwig 2006] and would help to structure models to the avoidance of errors.

   c) Identification and auditing of errors
A current research direction which continues to bear fruit is the automatic identification of spreadsheet errors and poor practice [Aurigemma and Panko 2007] [Hermans 2012]. A number of research and commercial tools have been developed which support the auditing of spreadsheets. This research will continue to bear fruit in the future and will need to focus upon how data is transformed and processed in the age of Big Data. For example capturing and auditing the data pipeline within a spreadsheet - how data is imported and cleaned within a spreadsheet is likely to be a source of potential errors and useful tools.

   d) Refactoring tools for spreadsheets
Having identified errors or quality issues it is common in programming development environments to provide tools to fix these issues. This maybe done either automatically or under the guidance of the user. While some research tools of this kind have been developed for spreadsheets [Abraham, Erwig 2007] they have not reached the end user. Further such tools should not only focused on the fixing of potential errors but on providing tools for transforming, or refactoring, the structure of spreadsheets. Potential refactoring’s might include “standardise cell formatting”, “split long formula” or “extract common calculation”.

   e) Mechanisms for Code Reuse
A major enabler of the development of complex programs are mechanisms for the reuse of complex functionality at various scales. Functions, Classes, Modules and Libraries of code all encapsulate specific functionality which may be reused again and again. Such mechanisms are not present within spreadsheets and their development would greatly enhance the scalability of complex spreadsheets and their rapid development by encapsulating key functionality. Challenges inherent within this are
how to identify such commonality and how to extract it naturally. Extraction of such commonality will require refactoring tools and would enable repositories of shared functionality.

f) Open source model repositories
In recent years large open source repositories of code have appeared for most programming languages (for example Github). Such repositories may contain small functional modules, for inclusion in other systems, or complete complex programs. Surprisingly, given the size of the user group, there are few similar repositories for spreadsheets. It is possible that this is a cultural challenge, “my spreadsheet is confidential” or that the quality of an average model is poor or that spreadsheets are simply too bespoke for reuse. Alternatively it may be that spreadsheets lack the clear point of abstraction at which common functionality can be clearly extracted and made reusable by many people. Should this prove to be the case then the provision of such an abstraction should prove valuable by enabling the sharing of complex functionality which may be peer-reviewed to the avoidance of error and improvement of quality.

g) Dealing with Data
One of the biggest challenges facing spreadsheets is how to enable working with large amounts of data which exceeds that which may be usefully stored within a traditional model (is it useful to refer or view row 99,999,998?). Combining such data into the spreadsheet will be essential for its future as the scale of data we interact with daily grows considerable and is no longer capable of being sensibly stored locally. It may be that as with existing tools for processing Big Data, rather than bringing the data to the algorithm; the algorithm should be sent to the data. This enables distributed processing of large volumes of data on compute clusters with algorithms prototyped locally then sent to the data to be processed on a cluster. Further improvements will be necessary in the way spreadsheets deal with data, particularly when it is unstructured and unreliable; dealing with inconsistent data in spreadsheets at present requires a sharp eye, tools to highlight inconsistencies are being implemented but will need to improve to support a larger scale.

h) Data analysis techniques
Spreadsheets are a natural environment for processing and analysing data. However the tools available to process this data are limited. Future spreadsheets will need to be augmented with machine learning techniques and provide integration with stronger data processing tools and ecosystems such as Python and the statistical language R. Whether such complex tools can be integrated in a user friendly way remains to be seen but perhaps Visual Basic for Applications is not the language for the future of the spreadsheet? Perhaps a new domain specific language for spreadsheets should be developed?

i) Version control and collaboration in spreadsheets
Version control systems have proven essential in professional computer programming which enable large teams to work together working concurrently editing the same codebase. Each team member is able to work on different features, changes can be put through an automated and peer-review process as changes are merged either automatically or line by line in case of conflict. Recent versions of spreadsheets now incorporate a “Track Changes” functionality which enables highlighting of changes made by different users along with a comment and review system. Such tools are a start toward this but more robust version control systems are required particularly in the context of increasingly online and collaborative method of working where large teams interact with a single document concurrently.

j) Interacting with spreadsheets in new form factors
The form-factor of computers we use has barely changed in the office environment for several decades. We still expect spreadsheet users to have a mouse, a keyboard and a large high resolution screen. These assumptions are increasingly being challenged by the rise of mobile and tablet computing bringing new modes of interaction such as touch, stylus and voice control. None of these modalities are widely used in spreadsheets. Similarly the reduction in screen resolution makes using and creating large spreadsheets very difficult, particularly given the large grid like matrix. Exploiting these changes effectively within the spreadsheet requires further research into end-user studies and design. Further into the future the rise of Virtual and Augmented Reality will provide the spreadsheet with the potential to move into a natural 3d environment. Quite how the spreadsheet will look in this environment is an open research question!
Conclusions:
Much of the recent research effort around spreadsheets focuses on the support of end users as they work with the existing spreadsheet model. Generally this is either through the identification of spreadsheet errors [Panko 2015] or with tools and techniques to support their understanding of the complexity of the spreadsheet. In the future research will be needed to adapt the spreadsheet to the challenges it faces as the environment in which spreadsheets are used changes. This will primarily be through an increase in the scale, variety and inconsistency of future datasets but also in the compute devices and online and more collaborative nature of future work. Evolving the spreadsheet to meet the challenges inherent in these changes provides an interesting research question, particularly if it is to be done without alienating existing spreadsheet users and the simple spreadsheet model which has been the underlying success of the spreadsheet over the past 30 years.

7. IMPLICATIONS FOR USERS

Finally we explore the implications of the drivers of change for the end users of spreadsheets. Future users should be increasingly technically literate having grown up in an online world using smartphones and tablets from a young age and being taught computer science in school [Brown et al 2014]. The authors believe that the following are some of the changes users should expect:

a) The same model
Fundamentally users should expect little change to the basic spreadsheet model which has stood the test of time and proved its versatility. Indeed changing this fundamental models is liable to encounter substantial opposition and the inertia effect of requiring retraining of a large number of users. This should not be surprising as the fundamental reasons for which spreadsheets are used daily are not going to change – the recording and manipulation of data, the creation of a calculation model.

b) Better Support
A growing body of research is focused on the support of end users of spreadsheets who should expect to see this work support them in their daily use of spreadsheets. From the identification of errors to the extraction of structure within spreadsheets these tools will increasingly aid users in working efficiently and more importantly safely with spreadsheet models. This will take the form of tools for gaining a better understanding of the spreadsheet model, perhaps through visualisation, and by tools for auditing existing models, suggesting fixes to identified issues. This trend should result in an improvement in the quality of spreadsheet models.

c) Easier ways of working with data
It is clear that the scale and variety of data which future users will seek to work with will continue to grow. Future spreadsheets must support users in this or they will lose their relevance to the next generation of users. This may be done by enabling spreadsheets to import large quantities of external data or by enabling the algorithms within spreadsheets to be executed on the cloud. At the least spreadsheets will require a more consistent data processing pipeline recording the steps taken to clean a data table or enabling better fusion or joining of datasets. As the number of datasets grows tools for interactively exploring such data will be provided as evidenced by Microsoft’s PowerPivot [Ferrari and Russo 2013]. Users should further expect to have a wider range of tools for analysing such data, machine learning is proving a key technique in analysing Big Data and will likely need to be incorporated into the modern spreadsheets expanding suit of tools.

d) Changing ways of working
Over the past 20 years we have seen the rise of laptop’s, smart phones, tablets and online collaboration suites. Each of these has changed the way we work and is having an effect on the way we interact with the humble spreadsheet. Future trends will likely see the rise of virtual and augmented reality which may impact upon the spreadsheet (which may no longer be limited to 2 dimensions!). In addition we will see a more collaborative online approach to working with spreadsheets with multiple authors and a more structured approach to model sharing and publishing. Perhaps we will see a rise of “open source” spreadsheet repositories or components.
Conclusions:
It remains to be seen whether these changes will be incorporated directly into spreadsheet software and the essential spreadsheet model or whether they will be evolved in external complimentary tools. It is likely that the future spreadsheet user will be offered a more extensive toolset to augment their basic spreadsheet, the challenge for such a user will be to navigate and learn these tools and to use them effectively. Conditions for using spreadsheets will also vary from different computing form factors to a more online and collaborative setting. This will present new opportunities and challenges for users who should still expect to find spreadsheets a key tool for working with data and modelling.

8. CONCLUSIONS

The success of the spreadsheet stems from its simple underlying model and the adaptability of that model to the everyday tasks which millions of people carry out. While this model and its implementations do suffer from some challenges (notably the hiddenness of errors) its continued success argues well for the future of the spreadsheet, particularly as this is driven by a continued underlying business need.

"...spreadsheets will always fill the void between what a business needs today and the formal installed systems..."
[Glass 2009]

The future of the spreadsheet will be driven by how they meet the Drivers of Change identified in this paper. These form challenges in two directions, changes in the nature of the tasks spreadsheets are used for and changes in the context in which they are used. Firstly the tasks spreadsheets are used for will change as the data we encounter daily grows in scale, heterogeneity and unreliability. We will also seek to use spreadsheets for different purposes in light of such data, either to explore it statistically or visually or to fuse or process it into new forms. Similarly new tools will be required such as machine learning as it becomes an increasingly effective method of gaining insight as the scale of data grows. Secondly the context in which spreadsheets are used will change either in the type of computing device used or toward a more social collaborative online spreadsheet experience. With an increased understanding of the risk of spreadsheets enterprise control (especially in a cloud based environment) will change the way we interact with spreadsheets with increased auditing and tool support. This does present opportunities for spreadsheets, for example the publishing of models to open source repositories or as the basis of a “Software as a Service” model.

Recent developments in spreadsheets in the past decade have primarily been augmentation of the successful core spreadsheet concept with advanced tools and functionality to support new needs of users. It remains to be seen whether simply augmenting the tools provided to users without adapting the core spreadsheet concept will be sufficient to ensure the success of the spreadsheet in the coming decades. This is an exciting and fruitful area of research as we seek to better understand and support millions of end users in their computing with spreadsheets.

9. ACKNOWLEDGEMENTS

The authors would like to thank ICAEW and Imperial College London’s Data Science Institute for hosting the Future of the Spreadsheet workshop and all of the conference participants for their insight and lively discussion, it is only with a strong expert community that the spreadsheet has a future.

10. REFERENCES


Alternative Spreadsheet Model Designs for an Operations Management Model Embedded in a Periodic Business Process

Thomas A. Grossman
Vijay Mehrotra
Mouwafac Sidaoui

University of San Francisco
2130 Fulton Street, San Francisco, CA 94117-1045, USA
tagrossman@usfca.edu, vmehrotra@usfca.edu, sidaoui@usfca.edu

ABSTRACT
We present a widely-used operations management model used in supply and distribution planning, that is typically embedded in a periodic business process that necessitates model modification and reuse. We consider three alternative spreadsheet implementations, a data-driven design, a canonical (textbook) design, and a novel (table-driven) technical design. We evaluate each regarding suitability for accuracy, modification, analysis, and transfer. We consider the degree of training and technical sophistication required to utilize each design. The data-driven design provides insight into poor spreadsheet practices by naïve modelers. The technical design can be modified for new data and new structural elements without manual writing or editing of cell formulas, thus speeding modification and reducing risk of error. The technical design has potential for use with other classes of models. We identify opportunities for future research.

1. INTRODUCTION
There has been much attention in the literature to the design of spreadsheet models for financial analysis. There are several methodologies (including SSRB 2016, FAST 2016, and Operis 2017) that are compared in [Grossman and Özlük 2010]. However, relatively little attention has been given to operational planning and analysis models, another area in which spreadsheets are widely used. This paper examines an operational model that has received scant attention in the spreadsheet literature despite being well-known in practice and in the operations management and operations research fields.

1.1 Overview
We describe the model in section 2. In section 3, we present three different designs to implement the model in a spreadsheet: a “data-driven” design, a “canonical” or textbook design, and a novel “technical” design that utilizes Excel Tables. We evaluate these designs according to a set of quality criteria, and also discuss the level of spreadsheet training and sophistication necessary for an operations analyst or manager to utilize each design. In section 4, we summarize the three designs and compare their suitability according to the quality criteria. We conclude and provide suggestions for further research in section 5.

1.2 Contribution
We use a model that is well-known in the operations management and operations research literature as our foundation. This paper presents different spreadsheet implementations of a single model and explores their provenance. We evaluate each on a set of quality criteria, and level of sophistication and training needed to effectively utilize it. We introduce a spreadsheet design which we see frequently with naïve modelers. This helps us to understand the root cause of certain poor spreadsheet practices that are known to be problematic. We diagnose the problem as a “data-driven” spreadsheet design. We contrast the data-driven design with the well-known
"canonical" design from the textbooks. We also present a new “technical” design that takes advantage of Excel’s Table feature to implement this model in a manner that makes even extensive modification very easy, without the need to write or edit any cell formulas whatsoever. This technical design has potential to be applicable to other classes of model, and we compare it to the data-driven and canonical approaches.

2. ANALYTIC SPREADSHEET MODELS FOR DISTRIBUTION AND SOURCING

We present a model that is representative of a class of distribution and sourcing problems. This model serves as the foundation for the rest of the paper.

2.1 Overview

We consider an analytic spreadsheet model for distribution and sourcing. In this model, a number of suppliers provide a product (such as raw materials, parts, or finished goods) to a number of destinations (such as factories, warehouses, or markets) that have a demand for it. Each destination must receive a specified quantity of the product (“requirements”). Each supplier has a maximum available quantity of the product (“capacity”). There is a cost associated with each supplier-destination combination.

The operations manager for the company that manages the destinations is interested in “costing out” different plans for sourcing the product from the suppliers. In assessing these sourcing plans, she may also be interested in evaluating other risk characteristics, such as excess supplier capacity available to meet increased demand levels.

2.2 Example: Canonical Design

In Figure 1, we present a specific example of this type of distribution and sourcing model, using the “canonical” design that is discussed in more detail in section 3.2. In this example, there are three destinations (listed in columns C-E and in columns J-L) and ten suppliers (listed in rows 6 to 15).

![Figure 1: Canonical Design for a Simple Distribution and Sourcing Model](image)

The model inputs are the cost coefficients (C6:E15), the supplier capacity levels (G6:G15), the requirements for each destination (C17:E17), and the sourcing plan (J6:L15). The model outputs are the total sourcing cost (K19), total supplied by each supplier (N6:N15), and total quantity delivered to each destination (J17:L17).

The cells that are grayed out correspond to supplier-destination combinations that are not available (due to reasons of quality, prohibitive cost, or other issues). The particular sourcing plan shown in Figure 1 has placeholder values of 1000 units of product from each supplier to each allowable destination.
The operations manager will experiment with different values for the sourcing plan in order to obtain insight into the sourcing plan she will ultimately commit to. This sourcing plan (J6:L15) is of great practical importance because it determines the quantity shipped from each supplier to each destination, the total amount supplied by each supplier, the total amount delivered to each destination, the total amount paid to each supplier, and whether a supplier is even selected to do business with the company.

We note that the model in Figure 1 represents a single time period. In the general case, the time period may be anywhere from a single day to an entire year. Of course, this business operates over time, and so in the next time period a new sourcing plan will be needed. Naturally, future time periods can have different requirements at the destinations, different capacities for the suppliers, or even different suppliers or destinations. Therefore, the model needs to be easy to modify so that it can be employed as part of a routine periodic business process.

2.3 Transportation Models Are Commonly Found in Spreadsheets

This model is an instance of what the operations management and operations research communities call the “transportation model” where “sources” (e.g., suppliers) provide resources to “sinks” (e.g., destinations) and each source-sink combination has an associated cost. Transportation models are applied to problems in manufacturing, logistics and distribution, personnel scheduling, vehicle routing, and inventory planning (see Ahuja et al., 1989 for additional applications).

Spreadsheets have proven to be very valuable tools for building and analyzing transportation models (LeBlanc et al. 2004, Shaoyun et al. 2011, Hong et al. 2012). The standard business school textbooks on spreadsheet modeling and analysis discuss transportation models, including [Ragsdale, 2012], [Winston and Albright, 2012], [Powell and Baker, 2013], [Gross et al., 2013], and [Asllani, 2014]. For the most part, textbook authors use the model as a basis for algorithms to identify an optimal plan, and pay minimal attention to issues of spreadsheet design.

3. SPREADSHEET DESIGNS FOR TRANSPORTATION MODELS

We present three designs that implement a realistic instance of a transportation model. We evaluate each design according to a set of quality criteria drawn from [Grossman et al 2011] including suitability for:

- Accuracy (of formulas; ease of coding)
- Modification (efficiency and accuracy) to enable reuse
- Analysis (use of model to generate business insight)
- Transfer (ease of understanding by other personnel)

We also discuss the level of spreadsheet training and technical sophistication necessary for an operations analyst or manager to utilize each design.

3.1 Data-Driven Design

We provide a (somewhat more elaborate) version of this type of sourcing and distribution problem to business school students every year in our courses at the University of San Francisco. The students’ assignment is to develop a high-quality analytic spreadsheet model, and then use their model to conduct analysis and deliver recommendations to management.

The students are given this problem in the form of a case study that includes a set of data. Figure 2 shows the data that is provided. As is typical in business, (but not in textbooks), the data is provided in a form convenient for the person (or system) that generated it, but it may or may not be convenient for development of a spreadsheet model.
Inevitably, a minority of our students scorn their professors’ guidance to pause and purposefully create a sound spreadsheet design. Instead, they implement the model by immediately writing cell formulas on the same worksheet as the raw data in Figure 2. Thus arises what we call a “Data-driven Design”, shown in Figures 3 and 4.
Regarding accuracy, this design has obvious risks of error during coding and is inefficient to code because the design does not allow a formula to be written once and copied. To see this in Figure 4, compare the formulas in cells I6 and I7, and in cells L4, L5 and L6.

Modification of a data-driven design is problematic. If Hickock begins to ship to Bone, this will lead to an additional row in the raw data below row 5. From here, the formulas for Total Shipped by Hickock (I5) and for Total Delivered to Bone (L5) must be manually adjusted. Adding additional suppliers or destinations would require more extensive changes, taking up more time and increasing the risk of errors. Analysis is subtly challenging, because when the analyst experiments with plan values in column F, it is not obvious why it impacts the totals in columns I and L. Transfer of the model to others is difficult because the design cannot be understood without careful examination of cell formulas. A data-driven design requires no training or sophistication on the part of the modeler; indeed, it is typically our weakest students who deploy this type of design.
3.2 Canonical Design

The canonical design uses a matrix to represent the units shipped (see Figures 1 and 5) and is presented universally in the textbooks referenced above. The matrix structure of this design intuitively displays the relationships between suppliers and destinations. Regarding accuracy, the coding is efficient and low-risk, because the model outputs of supplied (N6:N15) and delivered (J17:L17) are simple row and column sums, and the formulas can be written once and copied.

This approach requires that the ill-formatted raw data be converted to the matrix design. This requires manual coding of cell formulas. However, the cell formulas (Figure 6) are minimal “pointers” without any mathematical functions, and so their accuracy is easy to verify.

Depending on the structure of data for the new month, modification can be easy or challenging. If the supplier-destination combinations are the same, new raw data values can be pasted into the raw data module in columns Q and R of Figure 5, and model cell formulas do not need to be adjusted. If an existing supplier begins to ship to an existing destination that was previously prohibited, the structure of the raw data will be altered, requiring relatively simple modification to the structured data module by updating cell references to the new raw data module. All model calculations will be unchanged.
However, if new suppliers and/or destinations are added, the process for updating the model is cumbersome. For example, to add a new supplier to the model, the analyst must go through the following steps:

1. Insert a new row to the cost coefficient and sourcing plan matrix for the new supplier
2. Update the calculations in the “Delivered” row for each destination
3. Import the new version of the raw data
4. Update the links to the raw data in the cost coefficient matrix and to the Capacity column for existing suppliers and for the new supplier

The steps associated with adding a new destination are analogous to these. While these processes are straightforward, the work associated with them takes time and requires coding effort that is not risk-free.

The canonical design is very convenient for analysis. As the analyst experiments with different values of units shipped, the impact on Delivered and Supplied are respectively presented intuitively in the same row and column. The canonical design is easy to transfer because the row and column sums are intuitive. A modest amount of training and sophistication is required to discover (or know) the matrix design and how to code it.

3.3 Technical Design

In this section, we present a “technical” design that codes the model at a higher level of abstraction, leveraging the power of Excel’s Table feature. This allows for more efficient coding and dramatically more efficient model modification. The presentation assumes that the reader is familiar with the function of the Excel Table feature.

The technical design (Figure 7) comprises four Excel tables. The table labeled “Shipping Costs and Sourcing Plan” in B4:E17 (designated ‘Ship’ internal to Excel’s Table feature) contains information specific to each supplier-destination pair, which is shipping cost data, and units shipped (which is the sourcing plan). The table labeled “Supplier Table” in G3:I14 (‘SupSummary’ for the Table feature) contains information specific to each supplier, which is capacity data, as well as a formula that computes total Supplied. The Table labeled “Destination Table” in K4:M7 (‘DestSummary’ for the Table feature) contains information specific to each destination, which is data on requirements, as well as a formula that computes total delivered. The table labeled “Cost Table” in O4:O5 (‘CostCalc’ for the Table feature) contains information that is not specific to suppliers and destinations, which is the total shipping cost.

![Figure 7: Technical Design](image-url)
In Figure 7, blue cells are input data. Green cells are the sourcing plan. Orange cells contain cell formulas.

This approach requires that the ill-formatted raw data be converted to the table design. This requires manual coding of cell formulas in the blue input data cells. We note that the supplier-destination pairs in the raw data (Figure 2) are the same as the first three columns of the “Ship” Table, so the data in the table can be populated trivially with minimal pointer formulas similar to what was done in Figure 6. Likewise, the Supplier Capacity values in the “SupSummary” Table and the Required values for each of the Destinations are obtained using minimal pointer formulas.

To calculate the total amount shipped by supplier Georgican in cell I5, we need to sum up column E for those rows where column B is equal to “Georgican.” This calculation is performed using SUMIF (cell I5 of Figure 8). It is coded using a “table-style” formula that uses ‘@Supplier’ to indicate the value in the same row (row 5) Supplier column in the Table (i.e., “Georgican” in cell G5). It uses ‘Ship[Supplier]’ to indicate the column labelled Supplier in Table Ship (column B), and uses ‘Ship[Units Shipped]’ to indicate the column labelled Units Shipped in Table Ship (column E).

![Figure 8: Calculations in the SupSummary Table of the Technical Design. (To help the reader interpret column I, columns G and H show values although the cells actually contain formulas pointing to the raw data.)](image)

These table-style formulas appear cryptic until one learns how they work. In practice we never write the cell references manually, but instead click on cells or ranges as necessary, and allow Excel to generate the formula.

Notice that the cell formulas in each column are identical. This a powerful feature of Excel tables. The analyst need code a cell formula once in any cell of the column, and Excel automatically copies it to all other cells in that column of the table.

Regarding accuracy, coding is remarkably easy because references in cell formulas are generated automatically by Excel by clicking on the cell to be referenced, so the analyst need not attempt to write table-style formulas. Notably, the modest risk posed by copying formulas is eliminated entirely. As with the canonical design, the analyst must write simple pointer formulas to arrange the data; but of course this is minimally necessary for any design that is not data-driven.

The real significance of adding the abstraction of the Table is that the dimensionality and calculations of this model are entered once in the top row of the table and Excel automagically copies the formula down. Thus, modification is trivially easy. This is a significant prize. We
illustrate with an example, where an additional supplier called Paulucci and an additional destination called Duluth are to be added. See Figure 9.

Figure 9: Technical Design: New Data in rows 22:24

Excel Tables have the property that new data can be pasted (or drag-and-dropped) beneath a table, and the table expands automatically. Excel automatically generates any necessary cell formulas in columns that contain formulas. Excel even adjusts the formatting of the data automatically!

The analyst need only to select cells B23:D25 and drag them to cells B18:D20, then perform an analogous drag-and-drop (or copy/paste or cut/paste) for the other two tables, and the modification task is completed. (We manually entered values for Units Shipped in new cells E18:E20.) All the formulas update, all the calculated values update, all the number formats are adjusted. Not a single formula needs to be touched!

Figure 10: Technical Design: Modification complete without manual editing of cell formulas

The technical approach is useful for **analysis**, but has the same caveat as with the data-driven approach in that it lacks the row-and-column visual guidance of the matrix. This approach seems to be easy for **transfer**. Indeed, the use of the model requires only the ability to move data so that no programming is required for any new situation!

A high degree of training and sophistication is required to implement the technical approach. The analyst must be familiar with Excel Tables, be able to express the model in a more abstract Tables-based structure, and recognize how to adapt new data to the format required by the Tables.
Due to the ease of modification, it can likely be transferred to technically unsophisticated personnel.

4. Comparison of Designs and Recommendations

We summarize the three designs in terms of each of the quality criteria.

4.1 Accuracy -- Efficient and Correct Coding of the Model

The Data-Driven Design approach is fraught with the risk of errors due to the need to write multiple custom cell formulas by hand, especially as the number of suppliers and/or destinations grows. It does not require the use of pointer formulas to arrange the data, because the data is used as-is. The Canonical Design is the most intuitive approach and is relatively easy to build, with the primary effort being to organize the input data. It is necessary to write simple pointer formulas to arrange the data in the desired form. The Technical Design is extremely efficient and accurate. It is again necessary to write simple pointer formulas to arrange the data in the desired form.

4.2 Efficient and Accurate Model Modification in the Context of Operational Changes

The Data-Driven Design can accommodate new data easily as long as the structure of the inputs (valid supplier-destination combinations) does not change. However, such models are fragile, and any changes to supplier-destination combinations, new suppliers, or new destinations require a great deal of care and effort. The Canonical Design can accommodate new data – including new connections between existing suppliers and destinations – relatively easily. However, the process of adding new suppliers and/or destinations is somewhat cumbersome and risky. Because the Technical Design is agnostic as to the supplier-destination combinations, it can trivially accommodate anything, including new data, new supplier-destination combinations, new suppliers, and new destinations – all with no coding required.

4.3 Analysis and Generation of Business Insight

The matrix structure of the Canonical Design makes interactive “What-If” analysis easy and intuitive. Both the Data-Driven Design and Technical Design have a columnar structure that lacks the same visual intuition. In our experience this is not a problem for a sophisticated analyst. We hypothesize that one can use lookup functions to easily code a matrix-style report (ala the Canonical Design); this is a suitable topic for future research.

4.4 Transfer by the Spreadsheet Author for Use by Other Personnel

The major worry about transferring a model that follows a Data-driven Design is that any change whatsoever to the supplier-destination combinations will require manual programming, and may undermine the confidence of non-authors. In an environment where new suppliers and destinations are rarely introduced, the intuitive nature of the Canonical Design makes it easy to transfer to other personnel. From our experience, this representation of the problem in terms of supplier-destination pairs motivates user interaction with the model. On the other hand, in an environment where suppliers and/or destinations are frequently added, the Technical Design will be advantageous, as model users can easily be trained to update it themselves when structural changes take place, since one need merely move the data and need not touch any cell formulas.

5. Conclusions and Areas for Future Research

In this paper, we introduce the transportation model, which is used for a variety of practical operational problems and is often implemented in a spreadsheet. Using a simple but realistic illustrative example, we compare three different spreadsheet designs according to the criteria of accuracy, modification, analysis, and transfer, and we comment on the training and sophistication necessary to use each approach.
We believe that the flawed data-driven design has as its root cause a naïve approach to spreadsheet programming, where the author takes the structure of the data as a given, and starts to code directly without reflection about the implications of the resulting design. This perspective on data-driven design to spreadsheet development might be generalizable to other situations, and is an exciting area for further study.

We see a clear trade-off between the intuitive nature of the Canonical Design and the powerful flexibility that comes with the increased abstraction of our Technical Design. For relatively basic transportation problems such as our example problem, the choice between these two approaches depends largely on the sophistication of the model builders and end users and on the relative stability of the business environment. Each of these design approaches has its strengths and weaknesses.

Further research is required to understand the ease of analysis for the Technical Design, and in particular the possibility of coding a report that organizes the results into the intuitive matrix structure of the Canonical Design. We hypothesize that this can be done in a way that is amenable to accuracy, modification and transfer, but that remains to be demonstrated. This approach merits field testing or even empirical experiments; we will start that journey presently by teaching this approach to business students, which is likely to provide much insight.

The transportation model presented in this paper is a special case of a more general class of “network flow models” ([Ford and Fulkerson, 1962], [Edmonds and Karp, 1972], [Ahuja et al., 1989]) which have more complex “to/from” patterns and richer constraint sets. Network flow models have an even wider range of application than the transportation model. The insights in this paper for transportation models likely apply broadly to network flow models. Further research is needed to explore any limitations and required modifications or enhancements, particular in the context of additional conditions that are often referred to as “side constraints.”

It has been hypothesized (Siersted 2015) that the existing methodologies for financial planning models (notably FAST) have applicability to any analytical spreadsheet model. The operations management model presented in this paper can serve as a basis for future research to test this hypothesis. This line of research has potential to help us better distinguish between principles that are specific to financial planning models and those that are more generally applicable.

6. References


Electronic evidence and the presumption of reliability

By

Stephen Mason

Chambers of Stephen Mason, Langford, Bedfordshire

stephenmason@stephenmason.co.uk

The author, who is also the owner of the copyright, grants a Attribution-NonCommercial-NoDerivatives 4.0 International (CC BY-NC-ND 4.0) Licence to this article, and further grants arXiv.org a perpetual, non-exclusive license to distribute this article in the knowledge that a submission cannot be completely removed once accepted, and that arXiv.org reserves the right to reclassify or reject any submission.

ABSTRACT

In 1997, the Law Commission decided that there would be a presumption ‘that mechanical instruments were in order at the material time’ – a presumption that includes software code. This means that the prosecution can rely on evidence from software code being reliable and therefore does not require witnesses to authenticate the evidence adduced from software code. This has led to unfairness in legal proceedings. At the date of this paper, the presumption remains in place, and the legal profession continues to certify lawyers as being competent to practice without knowledge of electronic evidence.

There is a general concern amongst judges, lawyers and legal scholars that evidence in digital form is not to be trusted, given that it can be altered and manipulated with ease (although evidence in other forms, such a paper, are equally open to being fabricated, altered and manipulated). Some jurists have called for a UN Convention on matters relating to the authentication and admissibility of electronic evidence.¹

In England and Wales, the Law Commission created a common law presumption (that is, a presumption not created by Parliament) relating to mechanical instruments after the repeal of

¹ A Draft Convention on Electronic Evidence was developed and published in 2016: http://journals.sas.ac.uk/deeslr/issue/view/336/showToc.
s 69 of the Police and Criminal Evidence Act 1984 in 1999.2 The presumption included computers by implication (or more accurately, digital data). The Law Commission formulated the presumption as follows:

‘In the absence of evidence to the contrary, the courts will presume that mechanical instruments were in order at the material time’.3

Also, in criminal proceedings, s 129(2) of the Criminal Justice Act 2003 created a statutory presumption that a mechanical device has been properly set or calibrated:

129 Representations other than by a person

(1) Where a representation of any fact—

(a) is made otherwise than by a person, but

(b) depends for its accuracy on information supplied (directly or indirectly) by a person,

the representation is not admissible in criminal proceedings as evidence of the fact unless it is proved that the information was accurate.

(2) Subsection (1) does not affect the operation of the presumption that a mechanical device has been properly set or calibrated.

The aim of a presumption is to alleviate the need to prove every item of evidence adduced in court, or to reduce the need for evidence in relation to some issues.4 A presumption allocates the burden of proof between the parties.5 By way of a simple example, it is presumed that mechanical instruments were in order at the material time (or, in relation to software code, computers are reliable). This means a database is presumed to be reliable, and the content of

---

2 Section 69 ceased to have any effect by s 60 of the Youth Justice and Criminal Evidence Act 1999, and Schedule 6 also repealed s 69.


5 Colin Tapper, Cross & Tapper on Evidence (12th edn, Oxford University Press, 2010), 133 – 140 for a more detailed explanation.
the database is therefore reliable. This presumption is false, as illustrated in the case of *R v Cahill and Pugh*. So many issues have to be considered when analysing evidence from software code, that the presumption is demonstrated to be false – for which see the vignette ‘Business records’.

**THE PROBLEM WITH THE PRESUMPTION**

The problem with the presumption that a computer is deemed to be ‘in order’, or ‘properly set or calibrated’ is that software and the associated systems have become more complex. This means that it has become progressively more challenging to test software to reflect the way the users will use the product. This does not negate the fact that software written by human beings has always been – and continues to be – subject to errors.

The Law Commission commented about the practical issues of challenging the presumption at 13.14:

‘Where a party sought to rely on the presumption, it would not need to lead evidence that the computer was working properly on the occasion in question unless there was evidence that it may not have been – in which case the party would have to prove that it was (beyond reasonable doubt in the case of the prosecution, and on the balance of probabilities in the case of the defence).’

There are problems with judicial comments on this topic. First, there is no definition of what is meant by ‘in order’. Software might be ‘in order’ (whatever this means, but for the purpose of this article, the Law Commission and judges are taken to know what they mean) but not in the way an owner or user expects, because software can be both reliable and functioning consistently, yet perform functions without the authority or knowledge of the owner, and a third party can instruct software to do things that the owner neither authorises nor is aware of. Second, it will not always be obvious whether the reliability of digital evidence is immediately detectable without recourse to establishing whether the software code is not at fault. This leads to the logical conclusion articulated by Eric Van Buskirk and Vincent T. Liu:

‘The Presumption of Reliability is difficult to rebut. Unless specific evidence is offered to show that the particular code at issue has demonstrable defects that are directly relevant to the evidence being offered up for admission, most courts will

---


7 *Electronic Evidence*, xii – xiii.

8 As described in detail in *Electronic Evidence*, chapter 6.
faithfully maintain the Presumption of Reliability. But because most code is closed
source and heavily guarded, a party cannot audit it to review its quality. At the same
time, however, source code audits are perhaps the best single way to discover defects.
This difficulty gives rise to an important question: if a party cannot gain access to
source code without evidence of a defect, but cannot get evidence of a defect without
access to the source code, how is a party to rebut the Presumption? Rather than
wrestle with, or even acknowledge, this conundrum, most courts simply presume that
all code is reliable without sufficient analysis.¹⁹ (Footnotes omitted)
The party contesting the presumption will rarely be in a position to offer substantial evidence
to substantiate any challenge, because the party facing the challenge will generally be in full
control of the computer or computer systems that are the subject of the challenge, although it
is not always the case, given the promotion of cloud computing and recourse to sub-
contracting on a significant scale.

THE CURRENT PRACTICE IN CRIMINAL PROCEEDINGS

Prosecutors adduce electronic evidence in proceedings via s 117 of the Criminal Justice Act
2003, which provides a statutory exception to the hearsay rule for documents created in the
course of a trade, business, profession or other occupation. It seems, from the point of view of
practice, that defence lawyers in England and Wales regularly agree to the inclusion of
electronic evidence under this section on the basis of the presumption. However, the
provisions of s 117 do not remove the requirement that the evidential foundations have to
establish before the evidence can be admitted. It appears that prosecutors are aware of the
position, but defence lawyers do not, in general, appreciate this very important distinction.
Regarding the provisions of s 129(2) of the Criminal Justice Act 2003, the presumption that a
mechanical device has been properly set or calibrated arguably refers to devices such as
breathalyser devices, not computers or sophisticated networks. The commentary to s 129 does
not clarify the position:

‘432. This section provides where a statement generated by a machine is based on
information implanted into the machine by a human, the output of the device will
only be admissible where it is proved that the information was accurate. Subsection
(2) preserves the common law presumption that a mechanical device has been
properly set or calibrated.’

¹⁹ Eric Van Buskirk and Vincent T. Liu, ‘Digital Evidence: Challenging the presumption of
Additional confusion arises between the words to describe the presumption by the Law Commission that mechanical instruments were ‘in order at the material time’ and that provided by the explanation, and more crucially the statute, which uses the words ‘properly set or calibrated’.

THE LACK OF GUIDANCE

There is no authoritative guidance in relation to the meaning of the words ‘reliable’, ‘in order’, ‘accurate’, ‘properly set or calibrated’ or ‘working properly’ as variously used by judicial authorities and the language used in legislation in the context of digital data. It is possible to refer to system reliability, interpreted broadly, as a measure of how a system matches the expectations of the user. This view is problematic, because the expectations may be mistaken and can change arbitrarily, sometimes based on the user’s experience. Consequently it is arguable whether a more relevant test is whether the source of the data, which must come from a device or series of devices, was functioning consistently at the material time. However, even this is challenging, which is why it is probably better to fall back on the five part test to assess the authenticity of digital evidence, which departs from the concept of authenticity, so well known when testing physical objects, to the demonstration of integrity, trustworthiness and reliability. These tests are arguably more appropriate concepts to assess digital data, because digital data is derived from software and hardware, and placed before the user or viewer many times removed from their source: for instance, a transaction by way of an ATM (which may or may not be owned and operated by a bank) is routed from the machine via a third party communications provider to the back-end systems of the bank (or possibly a third party operating on behalf of a bank), before the transaction is entered into the bank’s system.

CONCLUDING COMMENTS

It has been proposed that the presumption created by the Law Commission that mechanical instruments – in particular that computers (now an out-of-date concept) are ‘in order’ – be reconsidered. The Institute of Advanced Legal Studies set up a Think Tank on Law Reform, and the proposals set out below regarding the presumption were submitted by the author for


\[11\] For a detailed explanation of the complexities of the ATM system, see When Bank Systems Fail Debit cards, credit cards, ATMs, mobile and online banking: your rights and what to do when things go wrong (2nd edn, PP Publishing, 2014), chapter 2.
further discussion with a view to submitting a more considered proposal to the Law
Commission in the light of the Commission’s 12th Programme of Law Reform:¹²

1. That consideration is given to more fully understanding the terms ‘reliable’, ‘in
order’, ‘accurate’, ‘properly set or calibrated’ or ‘working properly’ in the context of
digital data. For this purpose, it is wise to take into account the technical software
expertise that is readily available.

2. The defence are not in a position to demonstrate any defects that might be present
in relevant source code, or where the design of systems (that might also interact with
devices) may give rise to technical deficiencies that permit malicious third parties to
take advantage of poor quality software, or weaknesses in the design and installation
of systems and devices. This means the bar for raising doubts about the reliability or
otherwise of a computer, computer-like device or network must not be placed too
high.

3. That thought is given to a change in pre-trial criminal procedure, by placing a
requirement on the defence to alert the prosecution in advance of trial if there is to be
a challenge to the integrity and thus reliability and trustworthiness of any evidence in
digital format. Such a requirement will not prevent the defence from requiring more
detailed foundational evidence in circumstances where the evidence reveals
unexpected problems with the digital evidence at trial.

To achieve points 2 and 3, it was suggested that the following be considered:

(i) That the Criminal Procedure Rules be amended to incorporate a specific requirement that
the parties give active consideration to the integrity and thus reliability and trustworthiness of
evidence in digital format.

(ii) That a specific requirement is introduced to the Defence Statement, requiring the defence
to inform the prosecution when the authenticity of digital evidence is in question.¹³

(iii) That where the prosecution rely on the presumption that digital data (or evidence from a
computer or a series of networked devices) were in order at the material time, and the defence
raises a valid hypothesis respecting:

¹² IALS Think Tank Proposal (29 August 2013, revised 11 September 2013), no longer
available on-line.

¹³ With thanks to Sandip Patel of Furnival Chambers for further refinement of (i) and (ii).
(a) technical deficiencies, poor quality software, or weaknesses in the design and installation of systems and devices (or any combination thereof) that serves to undermine any evidence predicated on the presumption, or

(b) where technical deficiencies, poor quality software, or weaknesses in the design and installation of systems and devices either on their own or in combination with the actions of a malicious person could have subverted a system running on software (acting internally or externally),

then the prosecution do not benefit from the presumption.14

The author has previously argued that inadequacies in procedure, and by implication, presumptions, can cause unfairness.15 Surely it is appropriate for the authorities to reconsider the presumption in England & Wales in the light of the fact that the present formulation, ‘mechanical instruments were in order at the material time’, is not appropriate to the nature of evidence in digital form.

© Stephen Mason, 2017

Stephen Mason is a barrister and a Visiting Lecturer at the School of Law, University of Tartu, Estonia.

This article is adapted from Stephen Mason, ‘Electronic evidence: A proposal to reform the presumption of reliability and hearsay’, Computer Law and Security Review, Volume 30 Issue 1 (February 2014), 80 – 84

14 With thanks to Ross Anderson, Professor in Security Engineering at the University of Cambridge Computer Laboratory for the discussions regarding the formulation of (iii).

Mitigating Spreadsheet Risk in Complex Multi-Dimensional Models in Excel

Steve Litt
Indigo Sun, Inc.
36 Yardley Court Glen Rock, NJ 07452
sj@rumfox.com

ABSTRACT

Microsoft Excel is the most ubiquitous analytical tool ever built. Companies around the world leverage it for its power, flexibility and ease of use. However, spreadsheets are manually intensive and prone to error, making it difficult for companies to control spreadsheet risk. The following solution is designed to mitigate spreadsheet risk for a set of problems commonly addressed in a spreadsheet defined as “complex multi-dimensional models”: “Complex” referring to certain types of applications that require functionality such as sophisticated algorithms, challenging hierarchies and database write-back (i.e. planning, forecasting, etc.) and “multi-dimensional” referring to providing capabilities such as reporting, data input forms and ad hoc analysis on the different attributes associated with the resulting model. The solution is defined as a “PivotModel” because it works similarly to a PivotTable but is designed to leverage the robust capabilities of the Microsoft Excel platform.

1. MITIGATING SPREADSHEET RISK IN COMPLEX MULTI-DIMENSIONAL MODELS

Companies around the world leverage Microsoft Excel for its power, flexibility and ease of use. There are many different problems it addresses. One common problem is “complex multi-dimensional” models. The following is an example of a complex multi-dimensional model showcasing a P&L for a fictitious global Lighting Company.
There are a few areas in this example that help define “complex”. First is the reporting structure – notice the P&L rollup is not symmetrical, but instead ragged and unique to this company’s reporting needs. Next, it is evident that the Budget column has a need for users to manually input data. Also, for line items such as “General & administrative” or “Sales & marketing”, the detail costs might not be known for each office and product. For instance, this might be a cost only associated with Corporate and therefore requires an algorithm to allocate the cost down to each office and product to determine profitability.

The next component to this example helps define “multi-dimensional”. Figure 1 is a report that the Quality Lighting Corporation wants to create for all offices and products. In this report, Accounts are in the rows, Scenarios are in the columns and the report is run for Quarter One. Therefore, counting totals and subtotals, there are nine (9) offices and five (5) products. Since PivotTables are not designed to support this complex problem, users resort to building out the combination, 45 (9x5) spreadsheets to support this model.

Also, these 45 spreadsheets are just for one (1) report. If the users want another type of report (i.e. P&L over Time) then that requires another 45 spreadsheets. Also, if this company adds another office and product, we are now up to 60 (10x6) spreadsheets to support this view.

If we introduce Time (i.e. Qtr1, Qtr2, Qtr3, Qtr4 and Year) to this model, we potentially need 12,600 cells (ACCTS (14) x SCENARIO (4) x ORG (9) x PRODUCT (5) x TIME (5)) to calculate all the combinations of this model. And, sans allocations etc., 1,728 cells (ACCTS (9) x SCENARIO (2) x ORG (6) x PRODUCT (4) x TIME (4)) or 14% of this model is data. The rest, 10,872 or 86%, of this model are calculations. Making sure these calculations are correct is incredibly difficult and this is a very simple example. …Imagine a model that contains 10,000,000 (10^7) cells?

Given this magnitude of potential calculations, it is easy to surmise that 9 out 10 spreadsheets have potential errors [Panko, 1992] and the consequences can be extreme. There are numerous stories where spreadsheet errors have caused companies huge financial losses.

2. CURRENT STATE OF THE ART OF SOLUTIONS

The current method of solving these types of problems is to leverage Excel’s row, column, worksheet and workbook paradigm to build the desired multi-dimensional model. The model design is predicated on
being able to address the permutations of the members associated with each dimension. In the example above, we are dealing with 12,600 cells.

When creating a model, users are faced with the need to manually develop and enhance their spreadsheets. These labor-intensive solutions are prone to error and introduce the potential for spreadsheet risk. There are techniques that leverage more sophisticated methods [Bartholomew, 2016] like range names, array formulas, lookup functions and other techniques to help mitigate spreadsheet risk. Implementing these solutions involves a more experienced spreadsheet user but still have the potential for error.

Once a spreadsheet is complete and if/when an error occurs, it is extremely difficult to diagnose. As well, spreadsheets fall short in providing documentation that meet regulatory requirements. There are several tools in the market today [Schalkwijk, 2015] designed to facilitate a user’s effort it recognizing incorrect calculation logic to mitigate risk. These solutions, however, are utilized after the model is built. They reduce risk by searching through models and isolating on areas that meet criteria of a list of common problems they are aware of. Our solution is designed to programatically help in the initial model development, thereby mitigating, if not eliminating, the “human error” risks.

3. NEW ALTERNATIVE SOLUTION

The following is a new method in addressing complex multi-dimensional modeling in Excel. It is a programmatic approach to calculation development that mitigates the potential of human error and consequently, spreadsheet risk.

We call our solution a “PivotModel”. The reason we use the term PivotModel is to differentiate it as a more robust version of a PivotTable. However, the PivotModel is an all-in-one solution with characteristics that are also similar to Microsoft’s Data Models and Power Pivot.

For example, a Data Model provides the capability to integrate data from multiple tables, effectively building a relational data source inside an Excel workbook. The PivotModel acts as a meta data container that allows users to load disparate data from multiple tables without the need for SQL. The result is an excel-based workbook indexed to provide multi-dimensional slice and dice capability like a PivotTable.

It is similar to Power Pivot in that it provides a language to create calculations. For a PivotModel, the language is the functions and calculation capabilities found in Excel. Users find it easy to get up to speed on the rule logic because they are familiar with native Excel. For Power Pivot, it is a specific language called DAX that is relational-like and used to produce calculations.

3.1 Technical Overview

The PivotModel can best be described as a MOLAP solution in that calculations are pre-generated for fast query performance. It also handles complex calculations and read/write access to support Reporting, Analysis, Modeling and Planning applications.

ROLAP solutions like Tableau and others differ significantly in that the common data storage is relational and they do not pre-compute and store information. The ROLAP solutions are designed to handle Reporting and Analysis.
The PivotModel is designed to complement Microsoft’s existing toolset. For instance, the PivotTable handles rack’em stack’em aggregation to support analysis and data visualization. There are, however, a plethora of more complex business problems that these types of solutions don’t address.

For instance, consider an application that requires users to compare price-volume-mix to better understand product variances. Or, an application that handles reciprocal allocations that perform simultaneous equations to distribute indirect costs to profit centers. Or, a solution that models a leverage buyout (LBO) comparing internal rates of return from different what-if scenarios. These are the areas where Excel really shines.

By creating a solution that resides on top of the Excel technology, we not only tap into the power of Excel, but amplify its results by simplifying rule logic, ensuring calculation accuracy and providing multi-dimensional slicing and dicing capabilities.

There are three aspects to developing a PivotModel: Building a Model Structure; Creating Rule Logic; and Loading Data. Once complete, the user develops Views and/or Input Forms from the PivotModel for reporting, analysis, modeling and planning. This method differs from other technologies in that it approaches analysis from a “Top-Down” perspective instead of “Bottoms Up”.

Technologies like PivotTables, Power Pivot, Tableau, etc. focus on gathering data and merging it together resulting in a table that houses all the data and meta data in which to perform analysis. Thus, the concept of Bottoms Up because analysis is driven by the resultant table produced. Whereas with a PivotModel, users create the desired analysis first and then procure the required data to support it. This Top Down approach permits flexibility in modifying meta data and calculations as well as provides methods to integrate data not easily performed with SQL.

3.2 Building a Model Structure

The Model Structure includes dimensions, members of the dimensions and the member hierarchies associated with each dimension. For instance, utilizing our Lighting Company example, we have five dimensions: Accounts, Time, Product, Organization and Scenario. The following represents the members in that model structure.
Therefore, there are five dimensions in this model, Accounts has ten (14) members, Time has five (5) members, Product has five (5) members, Org has nine (9) members and Scenario has four (4) members. The number of cells in this simple model with data and calculations is 12,600 (14x5x5x9x4). Each of these dimensions has a hierarchy. For example, one could drill down on each of these dimensions to detail. For example on TIME, we could drill into “Year” which is comprised of Qtr1 thru Qtr4.

3.3 Creating Rule Logic

The following depicts the rules associated with the Accounts dimension of this model. The dimension and its members are in a hierarchy expressed in Column A. In Column B resides the data with Excel Rule Logic. And, in Column C, the Excel cell logic is displayed. Therefore, cell B5 (9,900.58) is computed as B3–B4.
In the Excel paradigm, every calculation references a cell or range of cells. Therefore, when reviewing the calculation it will refer to the Column and Row coordinates (e.g. B5 or R5C2). When dealing with large amounts of cells with logic, there is the potential of users mistyping or referencing the wrong cells in Excel.

From a modeling perspective, it is better to refer to the calculation in business terms. In this case, the business logic associated with this calculation is really “Net Sales”, which equals “Total Sales” minus “Discounts and Allowances”. Our application creates Rules instead of cell logic making the model easier to understand and simpler for users to ensure the model is computing accurately. Here are the Rules (12) associated with the Lighting Company’s PivotModel.

Rules 6-10 in Figure 5 reflect the same logic as the Excel calculations example displayed in Figure 4.
The Rule Editor acts as a calculation interpreter walking through the Rules in sequence and applying them to the PivotModel. The application utilizes Excel’s extensive library of functions (i.e. VLOOKUP, etc.) with the ability to parameter drive the rules with dimension members and/or provide additional logic for calculations such as time intelligence. The Column Type refers to whether the line is a Standard or Custom Rule or a Folder or Sub-folder.

The major difference is the Rules represent logic as it pertains to every combination within the model. Therefore, “Net Sales” is computed for every PRODUCT, TIME, ORG and SCENARIO. This dramatically reduces spreadsheet risk because one rule refers to numerous calculations in the resulting Excel model. For instance, this particular Rule represents 900 calculated cells (PRODUCT (5) x TIME (5) x ORG (9) x SCENARIO (4))

It is important to note that this represents the maximum number of calculations for this formula. However, in an analytical model like this, in the other dimensions, for instance, ”Year” in the TIME dimension, there is the potential of two calculations producing the same results. To clarify, in some situations, we could add up “Net Sales for the Year” (i.e. “Net Sales for Qtr1”+”Net Sales for Qtr2”+etc.) and it would equal the same results as if we computed “Net Sales” as the difference between “Total Sales for the Year” minus “Discounts and Allowances for the Year”.

The scenario above produces the same results, however, in different situations, this might not be the case. Consequently, it is imperative that the right Rule take precedence or your model may produce incorrect results. Using our “Net Sales” and “Year” example, if these Rules took precedence in the calculation for the “Var %” member in the SCENARIO dimension, it would produce the wrong results because the Year rule would add up the %Var for all Qtrs which is incorrect.

The solution allows users to control the calculation precedence by providing the ability to control the sequence in which the calculation logic takes place. Using the standard Excel framework, users must determine the precedence of every calculation (12,600 cells) in their model. Anyone familiar with Excel knows how easy it is to err by dragging forward a rule (i.e. copy and paste) to additional cells when it is not the right calculation.

Excel is a very powerful and flexible solution. However, it is error prone and manually intensive because users have to perform repetitive rule logic and determine proper calculation precedence. Our application is designed to programatically take care of this repetitive logic development and calculation precedence to mitigate spreadsheet risk.

The goal is to provide functionality that matches or exceeds the capabilities inherent in Excel. Therefore, at a minimum, anything you can do in Excel, you should be able to do in the solution. For instance,

- Filtering – being able to apply logic to specific members in the PivotModel (ie. Only apply a calculation to specific members)
- Cross-Dimensions – being able to apply logic across dimensions (i.e. Each Product’s Sales as a % to Total Sales for All Products)
- Time Intelligence – being able to create calculations for time (i.e. Year-to-Date, This Year vs Prior Year, etc)
- Calculate and Store – able to produce logic that results in data giving the user the ability to override the results (create a forecast from historical data and allow the user to override)

The final results reside completely in Excel. There is no multi-dimensional data source outside the Excel framework that gets queried and then displays the data in the spreadsheet.
3.4 Loading Data

Every model stores base level data. For instance, Column C in Figure 4 displays Excel rule logic for the cells in Column B. The other cells in Column B are considered Data.

The model structure acts as a container. Users can load data from multiple data sources to different parts of the PivotModel. For instance, we may have Actual and Budget data. The example below is Actual data from a back-end ERP in the following format, while the Budget data may come from other data sources.

![Figure 6: Sample Load File for Actual Data from the Company ERP](image)

Here is an example of another data load file type. This one is specific for Europe's Budget Data with the Accounts going across the columns. The data can be imported and exported in different formats.

![Figure 7: Sample Data Load File for Europe’s Budget Data](image)
Both Figure 6 and Figure 7 are examples of different data file formats that a user can use to load data into the PivotModel. The PivotModel acts as a container where users load data from different disparate data sources. Data can be imported if it is associated with unique members of each dimension. Therefore, users can load data from multiple files. There is no need to consolidate all the data into one data set before viewing the data as required in a PivotTable. This reduces the need for advanced SQL skills to analyze information.

### 3.5 Views & Input Forms

The application provides reporting and ad hoc capabilities to view the multi-dimensional data once the PivotModel is complete. If the PivotModel resides on a universal drive, multiple users can access it for reporting and ad hoc analysis. The following is a sample view.

![Figure 8: Results in Sample PivotModel View](image)

**Figure 8 Captions**

- **Custom Views -** The solution provides developers with the ability to build custom views against the PivotModels. It supplies designers with menu controls to give users seamless access to different PivotModel reports.

- **Write-Back (for Planning, Forecasting and What-if Analysis) -** At the solution core is Excel. Therefore, users can write back any type of data (numeric, text, dates, url, etc.) to the PivotModel.
It is possible to design robust driver-based solutions for modeling. The following depicts how the PivotModel is accessed via their Excel front-end.

Once the PivotModel is developed, users can pull data to view results. But they also can “write-back” to the PivotModel. For instance, User1 is responsible for Business Unit 1. Corporate needs User1 to provide a Budget for next year. User1 brings up a view and then enters their budget and hits “send” and the data is written back to the PivotModel.

- **Ragged Hierarchies (for Financial Reporting)** - Users need the flexibility to build any type of financial or management report. The solution allows users to create reports the way users want to see them without constraints commonly found in tools like PivotTables and Tableau.

- **Pivot Data (for Slicing and Dicing)** - The solution allows users to perform multi-dimensional analysis to slice and dice and drill down into pivot table detail.

- **Utilizes Excel capabilities** - If you can do it in Excel, you can do it within the application. Leverage all of Excel's functionality - fonts, data types, themes, conditional formatting, functions, macros, add-ins, sql database access, PowerPivot. All these capabilities are available within the framework.

- **Drill-down to Relational detail** - If a PivotModel is developed off a relational table or PivotTable, the solution allows users to drill down with a focused query to the supporting table detail. These tables can be linked to outside data sources via traditional Excel functionality or third party tools that access proprietary data sources.

- **Diagnostics** - The solution allows users to trace through the PivotModel by the formulas that produced the result. The diagnostic tool gives users detail explanation on how each cell was computed. This is explained further in the next section.

- **Link PivotModels to PivotModels** - The application provides the ability to link data between different PivotModels. This simplifies development of complex business solutions. For instance,
create consolidated budget pivot tables derived from a multitude of different departmental budgets.

3.6 Diagnostics & Documentation

The solution captures the interrelationships of the algorithms to provide methods in which to diagnose results. The following is a sample View for our Lighting Company. We are at a consolidated level and recognize a -0.6% for “% Var of Net Sales” and want to investigate the details.

Figure 9: Sample PivotModel Query at Qtr1 for Total Org and Total Product

Double-clicking the cell drills into the detail taking the user to a separate worksheet to analyze the calculation logic that produced the results. The user can double-click on cells to step through the logic for each calculation to determine how it was derived. This capability is best described as a multi-dimensional Trace Precedence where users step through the business logic to determine how certain cells are computed.

The following shows a user drilling into the formula for %Var of Net Sales. First drilling into the numerator of the Net Sales’ formula (Actual – Budget). Then drilling into Budget for Total Product, then drilling into Outdoor products to get to Total Organization, then drilling into Europe from the Total Europe/AsiaPac, then drilling into Europe’s Net Sales to see the detail data supporting Net Sales: Sales – Discount Allowances.
The solution also houses details of the model design to support internal Business and IT needs as well as regulatory requirements. The following are excerpts from the documentation.

In summary, we mentioned 12,600 cells in which 1,728 cells (14%) were data and the remaining 10,872 cells (86%) were calculations. Since the application programmatically develops all the calculations, there are only 12 rules to manage the logic of 10,872 cells dramatically simplifying the development and management of complex multi-dimensional models.

4. THE SOLUTION COMPARED TO OTHERS
There are many solutions in the market, both from a design and product perspective, that are focused on issues associated with complex multi-dimensional modeling. For example, in “How do you know your spreadsheet is right?” [Bewig, 2005], there are step by step processes one can go through to minimize spreadsheet risk. Interestingly, in one of the diagrams, there is a “dependence graph” reflecting the concept of Rules. This approach, as well as others, addresses the problem in a similar manner by manufacturing a solution assuming the Excel Row, Column, Worksheet, Workbook paradigm as its baseline. The solution starts with a conceptual framework first (Build Model Structure, Create Rules, and Load Data) then develops the model leveraging the Excel paradigm.

There are also tools in the market designed to recognize incorrect calculation logic after the model is built. Our application is designed to programmatically help in the initial model development, thereby “proactively” mitigating “human error” risks before they occur.

Finally, it is also important to note that empirical studies [Beckwith L., Cunha J., Fernandes J.P., Saraiva J., 2011] indicate model-based spreadsheets improve end user productivity. Model-Driven Engineering (MDE) techniques assist in end user effectiveness and efficiency. Therefore, benefits can be realized leveraging an MDE approach to spreadsheets.

Although the focus of the PivotModel is complex multi-dimensional modeling, a by-product of the application handles reporting and analysis. There is an extensive amount of Excel reporting tools in the marketplace which we don’t consider ourselves as a replacement. As mentioned earlier, the intention of the PivotModel is to complement the Microsoft tool set. Therefore, our reporting and analysis is associated specifically with the PivotModel construct but the results of the PivotModel can be pushed to other reporting technologies if desired.

5. CONCLUSION

In this paper, we described our solution for mitigating risk in complex multi-dimensional models. This programmatic approach is unique and proactively addresses the issues of human error in spreadsheet model development.

There are tools in the marketplace designed to recognize incorrect calculation logic. These tools, however, work after a model is built. Our solution is designed to programmatically help in the initial model development. Consequently, the solution works to mitigate “human error” before it occurs.

Our solution aligns more with prior Model-Driven Engineering (MDE) concepts in that it approaches the problem by mitigating risk through design. However, these other alternatives begin with the premise of the Row, Column, Worksheet, Workbook paradigm as their baseline. Consequently, they approach the problem leveraging existing functionality like Range Names, Formula arrays, lookups, etc. to produce a resolution.

Although our solution may eventually leverage similar functions, it begins with the abstract concept of building a model outside the spreadsheet paradigm. Therefore, the framework of developing a solution entails three distinct components: Building a Model Structure, Creating Rule Logic and Loading Data. Once these components are defined, the solution dynamically generates the Excel-based data store called a PivotModel for users to slice and dice results.

Complex multi-dimensional solutions have been available for over 35 years. These types of applications require experienced developers with both a technical and business acumen. The challenge in bringing this technology to the desktop is figuring out ways in which to teach analysts how to utilize these tools. If we
can create this new breed of analyst then we will see a transition from their current data cleansing and report creation role to one focused on data analysis and business performance.

6. REFERENCES

[Panko, 1992], Panko, Raymond “What We Know About Spreadsheet Errors”, Raymond R. Panko University of Hawai‘I, College of Business Administration, http://panko.shidler.hawaii.edu/SSR/Mypapers/whatknow.htm, 10:30am 4/24/2017


[Bewig, 2005], Bewig Philip L “How do you know your spreadsheet is right? Principles, Techniques and Practice of Spreadsheet Style”, EuSpRIG July, 2005

ABSTRACT

Auditors demand financial models be transparent yet no consensus exists on what that means precisely. Without a clear modeling transparency definition we cannot know when our models are ‘transparent’. The financial modeling community debates which methods are more or less transparent as though transparency is a quantifiable entity yet no measures exist. Without a transparency measure modelers cannot objectively evaluate methods and know which improves model transparency.

This paper proposes a definition for spreadsheet modeling transparency that is specific enough to create measures and automation tools for auditors to determine if a model meets transparency requirements. The definition also provides modelers the ability to objectively compare spreadsheet modeling methods to select which best meets their goals.

1 INTRODUCTION – STATE OF THE ART

Transparency became a major topic after the accounting scandals of the 1990s moved the United States congress to introduce the “Corporate and Auditing Accountability, Responsibility, and Transparency Act of 2002” which later became known as the Sarbanes-Oxley Act of 2002, or SOX [Lasher 2008]. Major global financial modelling standards continue to maintain a focus on transparency. For example the T in FAST stands for transparent [FAST 2016]. The words “transparency” and “transparent” appear numerous times on Corality’s SMART webpage. Phrases like “improving the transparency” [H.R.3763] [FTC 2016], “enhance transparency” [FAST 2016], “higher levels of transparency” [Schnackenberg, 2009] imply financial transparency is quantifiable yet none provide any transparency measures.

2 WHY MEASURE TRANSPARENCY

With transparency measures we can objectively identify:

- Opaque methods to exclude from modeling standards
- Opaque model sections to correct prior to model validation
- Models meeting transparency requirements and are thus ready for model validation
- Methods improving transparency to adopt to reduce model validation effort

Measures open the possibility for automation tools to facilitate transparency audits.
3 TRANSPARENCY DEFINITION

To develop automation capable of quantifying transparency we must fully understand what Excel model transparency means. In looking at the plethora of researcher definitions, “The common thread holding most definitions of transparency together is the notion that information must be disclosed to be transparent” [Schnackenberg, 2009]

Merriam-Webster Dictionary also provides several transparency definitions including “easily understood”. “Understandability” and “transparency” are linked with differences. To demonstrate a difference assume a model discloses all information and relationships used in a leveraged buyout model. According to Schnackenberg’s “common thread” definition, the calculation is transparent but a toddler would have no understanding of how it works or what it means. Understandability is linked to reviewer abilities. This proposal removes the reviewer abilities variable by assuming all reviewers have requisite skills. For more on understandability in the Excel context see “Measuring Spreadsheet Formula Understandability” [Hermans 2012].

Two other Merriam-Webster definitions complement Schnackenberg’s “common thread”:

- Easily detected or seen through
- Characterized by visibility or accessibility of information

The key phrase “accessibility of information” is a good starting point. To clarify it we can borrow a tool from Information Engineering designed to turn high level, vague concepts into more actionable components: functional decomposition [Marin 1989][Sage 1991].

Let us start by decomposing “accessibility” to “ease of access”. “Ease” can be further decomposed to “effort required,” thus “accessibility” becomes “effort required to access.”

To access something we must be able to detect it first and so “ease access” incorporates “easily detected” and what we are trying to detect and access is “information”. “Information” in an Excel model context is cell values. Each cell displays a value. Each cell’s value is a piece of information. The difference between data and information is data is a raw value and information adds meaning and context [Zins 2007] [Doyle 2014]. If a cell displays only raw data then for it to be information we must also find its label and that label must be sufficient to provide meaning and context. Thus, the result of our functional decomposition exercise is:

Effort required to access cell values and value labels.

That adequately defines cell surface level transparency but transparency as expressed in the definition phrase “seen through” infers a cell’s sources, values from which a cell derives its value, must also be accessible. Thus, a proposed complete model transparency definition is:

Effort required to access a cell’s surface and source values and value labels.
4 ESTABLISHING MEASURE

This paper proposes measuring effort. Merriam-Webster’s thesaurus lists several synonyms and related words for “effort” which include “work” and “power”. In physics power is the rate of doing work. In this case we need to quantify the rate of accessing cell values and labels.

4.1 Transparency Units of Measure

Excel provides several means by which we can find a cell’s source values. They include:

- Highlight a reference and press F9 to see its value
- Select a named reference from the names drop-down to navigate to its location
- Click a formula in the formula bar to see all local reference locations
- Use menu option FORMULAS > Trace Precedents to point to local references.
- And more

Each of these methods is trivial by itself and so this paper proposes assigning each a work unit of one step where a ‘step’ involves something other than just looking such as mouse clicks or keyboard entries. Sometimes we must repeat or combine methods to display a cell’s source values and labels. Each repetition is a step. Each additional method is a step. Thus, a cell’s proposed transparency measure is the minimum number of steps required to access all of a cell’s source values and labels. The proposed unit of measure is steps from transparency.

4.2 Transparency Terminology

To simplify discussions this paper proposes the following terminology:

**Labels**

This refers to that which provides meaning and context required to elevate raw data to information [Zins 2007] [Doyle 2014]. Labels can be placed in cells, data validation input messages, cell comments, documentation pages, external documentation or any other means accessible to inspectors and linked to values.

**Immediate Vicinity**

This indicates a group of cells can be displayed simultaneously within a monitor’s window. This will vary by content, monitor resolution and use of “freeze panes.”

**Surface Level Transparency**

This refers to values and labels viewable on the screen without having to select a cell.

**Source Level Transparency**
This refers to those references and literals from which the inspected cell derives its value. This only applies to cells with formulas. In the example above right we have selected a cell with a formula in which \texttt{A1} and \texttt{A3:B5} are source references, \texttt{2} and \texttt{FALSE} are source literals.

**Transparent**

A cell is transparent if no steps are required to see its surface values and labels as well as its source values and labels. Transparent is zero steps from transparency.

**Translucent**

A cell is translucent if any steps are required to find its values and labels as well as its source values and labels. Translucent is \(n\) steps from transparency.

**Opaque**

A cell is opaque if we cannot access either a cell’s value, label, source values, or source labels.

### 4.3 Convention

To keep measurement quantities aligned with the concept of transparency this paper proposes expressing steps with negative values. Thus, the measure of transparency for any given model reference, function or formula is negative one times minimum steps required to find source values and labels. In this convention:

- Transparent: 0 steps from transparency is completely transparent
- Translucent: \(-#\) steps from transparency is less transparent
- Opaque: \(-\infty\) steps indicates source values or sufficient labels are inaccessible

### 5 LABELING REQUIREMENTS

Occupied cells contain values and labels. Labels provide meaning and context for values. Label placement is subject to community standards while label content is subject to value type. Values can be categorized into the following types:

- Quantities
- Dates, Times and Durations
- Flags
- Identities
- Attributes

This paper proposes the following label content requirements based on value type.
5.1 Quantities

Quantities are magnitudes expressed as a number and reference. The reference includes a type/kind and unit [JCGM 2008]. In Excel terms, a cell containing a quantity value also requires labeling that includes type/kind and Unit.

<table>
<thead>
<tr>
<th>Subject</th>
<th>Examples</th>
</tr>
</thead>
<tbody>
<tr>
<td>Quantities</td>
<td>Monies, periods, dimensions, etc.</td>
</tr>
<tr>
<td>Types/Kinds</td>
<td>Receivables, debt term, length, mass, area, volume, etc.</td>
</tr>
<tr>
<td>Units</td>
<td>USD, months, employees, kg, cm, g, etc.</td>
</tr>
</tbody>
</table>

Units are also known as Units of Measure or UOM. Currency UOMs can be conveyed through formats that include currency symbols.

5.2 Dates, Times, and Durations

Dates and times mark when events occur or occurred. Durations are time quantities. Both are expressed in similar formats. These formats provide the UOM for various portions of the value. Because there are numerous formats their label must include their format as well as their subject.

<table>
<thead>
<tr>
<th>Subject</th>
<th>Examples</th>
</tr>
</thead>
<tbody>
<tr>
<td>Dates and times</td>
<td>January 1st, 2000; 12/1/2010; 11:00 AM; 01:23:14, etc.</td>
</tr>
<tr>
<td>Types/Kinds</td>
<td>Model Start, Debt Term, Period End, etc.</td>
</tr>
<tr>
<td>Formats</td>
<td>mmm dd yyyy, mm/dd/yyyy hh:mm, hh:mm:ss, etc.</td>
</tr>
</tbody>
</table>

5.3 Flags

Flags are Boolean values or switches. A flag's label must include the question the flag answers.

<table>
<thead>
<tr>
<th>Subject</th>
<th>Examples</th>
</tr>
</thead>
<tbody>
<tr>
<td>Flags</td>
<td>True or False, 0 or 1, and Yes or No</td>
</tr>
<tr>
<td>Questions (“?” implied)</td>
<td>Dued, Effective, Expired, etc.</td>
</tr>
</tbody>
</table>

5.4 Identities

Identities are names, numbers, or codes that uniquely identify an entity. An identity label must include the subject.

<table>
<thead>
<tr>
<th>Subject</th>
<th>Examples</th>
</tr>
</thead>
<tbody>
<tr>
<td>Identities</td>
<td>Employee IDs, Country names, Account Numbers, etc.</td>
</tr>
<tr>
<td>Types/Kinds</td>
<td>Employee, Customer, Account, etc.</td>
</tr>
</tbody>
</table>
5.5 Attributes

Attributes are non-numeric object properties. An attribute's label must include its subject.

<table>
<thead>
<tr>
<th>Subject</th>
<th>Examples</th>
</tr>
</thead>
<tbody>
<tr>
<td>Attributes</td>
<td>Variable Rate, Red, Round, Sour, Rough, etc.</td>
</tr>
<tr>
<td>Subjects</td>
<td>Loan Type, Color, Shape, Taste, Texture, etc.</td>
</tr>
</tbody>
</table>

6 EXAMPLES

This paper provides the following examples to promote deeper understanding of what is meant by transparency in the Excel modeling context. These are only examples.

6.1 User Inputs/Assumptions

In the example at right users make entries in the “Value” column. User entries have no formulas; thus, only surface level transparency is evaluated. Each input is labeled with Type/Kind and Units/Formats within the immediate vicinity. These are totally transparent.

6.2 Data Imports

In the example at right a dataset has been imported. The column heading provides Type/Kind labeling. Above quantity columns are units of measure/format labels. In this example the surface level is totally transparent.

Data imported using MS Query exposes its source by right clicking in the data and selecting Table > External Data Properties > Connection Properties (icon) > Definition (tab) and examining “Command Text:” This could be considered one step and applying to the entire table.

Data imported using Power Query exposes its source by clicking in the data then right clicking on the data’s query in the Workbook Queries panel and selecting Edit. This could be considered one step and applying to the entire table.

6.3 Literals in Formulas

We often find values expressed as constants embedded in formulas. In the example =A11 * 12 there is a literal: 12. We have no idea what 12 is other than a number. We do not know if it is a dozen, 12 inches in a foot, 12 months in a year, or something else completely. This source value is opaque.
6.4 Hidden Cells

A value that cannot be inspected is opaque. If a cell’s value can be revealed by authorized inspectors using other means then those steps must be counted in the cell’s transparency measure.

6.5 Error Cells

Cells displaying errors have inaccessible values and are opaque. Error values include:

- #DIV/0!
- #N/A
- #REF!
- #NAME?
- #VALUE!
- #NUM!

An exception is when errors are incorporated into downstream calculations as opposed to errors simply needing correction.

6.6 Unconstrained Indirect Reference

A cell’s value that is derived from Excel’s INDIRECT(), OFFSET(), LOOKUP(), VLOOKUP(), HLOOKUP(), or INDEX() functions that is not restricted to a specific cell range with appropriate labels is opaque because it is possible for the reference to point to cells with no value and/or label.

With care it is possible to make indirect reference functions translucent. The example at right provides model scenarios which can be selected via drop-down in B20. In cells B21:B23 is this formula:

\[ \text{=VLOOKUP( [@Label], INDIRECT( OFFSET( [#Headers], 1, 1, 1, 1)), 2, FALSE)} \]

- OFFSET() is constrained to a single cell relative to the table’s headers.
- INDIRECT() is constrained to a list of data validation values.
- VLOOKUP() is constrained to only what matches the current row’s label in either of the scenario tables.

These constraints limit these indirect reference functions to cells with accessible values and labels.
6.7 Literal Constants in Functions

In the example at right 3.5 is a literal constant. Literal constants expose their values (3.5); thus, there are no steps required to find them. Finding their labels, which tell us what their value means, may require many steps.

When we use a literal constant in a function, the function may provide a parameter label sufficient for identifying what the literal is. To display the function’s parameter labels we can double click the formula's cell or click the formula in the formula bar. Both methods expose a 'tooltip' below the cursor (red circle).

In this example we know 3.5 is a Rate. Rate is insufficient as it is lacking a subject (Interest) and a more meaningful unit of measure (APR). To find these required labels we must either double click PMT to bring up the function's help, or click the fx icon to display the Function Arguments dialog (shown right).

Sometimes parameter labeling is too vague. In the example we know 6 is the number of periods but we do not know if that is in weeks, months, quarters, etc.

Sometimes functions provide no meaningful labels at all. In such cases the literal is opaque.

A literal constant in a function's transparency is based on the function's parameter labeling being sufficient. For automation we can catalog Excel functions used and classify each parameter as sufficient or not. Thus, a literal constant in a function's transparency is -1 if the tooltip label is sufficient, -2 if we need to use the function’s Help or Function Argument’s dialog, or opaque if the function’s labeling is insufficient.

6.8 Cell/Range References

At right is an example model section. Cell B1 is properly labeled; thus, its surface level transparency is 0 (completely transparent). If we select cell B1 its
Formulas require us to add B1’s source level transparency by clicking anywhere in the formula bar. Excel color codes and highlights the cell references. Because everything is in the immediate vicinity we can see:

- The range A3:B5 comprises a list of labeled values.
- Cell A1’s value is “Initial Investment” which is a label
- The literal “2” is identified by the formula tooltip as the column index number
- The flag “FALSE” is labeled “[range lookup]” which is inadequate labeling. Adequate labeling is found by clicking the \( \text{fx} \) icon and reading the Help text.

This example is -2 steps from transparency because a single click in the formula bar exposes all source values and labels (1\(^{\text{st}}\) step) except the flag’s label located in the help text (2\(^{\text{nd}}\) step).

### 6.9 Named Range References

A named range is a defined name containing no functions or operators. It may contain a literal constant or cell/range reference. A name is a label and if it meets all labeling requirements eliminates the need to find a named range reference's label.

At right is an example using four named ranges in a formula placed in B1. This example is completely transparent because all surface values and labels as well as all source values and labels are visible simultaneously within the immediate vicinity.

If the named references were not in the immediate vicinity we could navigate to each named reference by using the Name drop-down listbox located left of the formula bar, or F5, then selecting the name from the list. This adds one step to each reference in which case B1’s transparency would be -4 steps from transparent (0 for B1’s surface level transparency plus -1 step to find each of four named reference + 0 steps for each reference’s surface level transparency).

### 6.10 Structured References
A structured reference is a type of dynamic named range generated automatically with tables. At right is a small table. A cell in the Net Income column is selected. It contains the formula:

\[ =[@\text{EBIT}] + [@\text{Tax}] \]

\[[@\text{EBIT}]\] is a structured reference that, like a cell reference, points to a location. While cell references point to locations in worksheets, structured references point to locations within tables. Cell references use worksheet names, column letters and row numbers as their name. Structured references use table names, column headings and special named regions as their name.

In this example, our selected cell has a transparency of 0 steps from transparent because \[[@\text{EBIT}]\] and \[[@\text{Tax}]\] are in the immediate vicinity along with required labeling.

7 TIMING

Transparency is only important when inspecting a model. Not all model users have the need, desire, time, skills, or authorization to appropriately inspect a model. If model information needs to be hidden for purposes of confidentiality or aesthetics the model is still ‘transparent’ if all pertinent source information is revealed when those qualified and authorized inspect it.

8 IMPLEMENTING THIS NEW METRIC

This paper proposes to implement this metric by assuming each cell is transparent and add to it associated component transparency measures. To describe the processes this paper uses pseudo code. Pseudo code provides human readable automation detail.

8.1 Cell Surface Level Transparency

Cell surface level transparency looks only at what is displayed in a worksheet cell. We can skip empty cells. Most model methodologies have regions set aside for labels which we can eliminate from scrutiny. Any cells displaying numeric values, regardless of location, must be measured.

If sufficient labeling not found then Transparency = \(-\infty\)

Else Transparency = Transparency – steps to find labeling

8.2 Cell Source Level Transparency

Cell source level transparency looks at what is inside a cell’s formula, thus, this only applies to cells with formulas.

For each reference in a cell’s formula

If reference is a literal then

If function’s parameter labeling insufficient then Transparency = \(-\infty\)

Else Transparency = Transparency – 1

Else
8.3 Cell Transparency

Cell transparency is the sum of its surface and source levels. If any level is opaque, the entire cell is opaque.

\[ \text{Transparency} = \text{Cell Surface Level Transparency} + \text{Cell Source Level Transparency} \]

8.4 Formula Transparency

A formula’s transparency is the host cell’s transparency.

8.5 Calculation Chain Transparency

A single result may include a set of formulas and references chained together. To calculate the chain’s transparency we must total the transparencies of all cells in the chain. One way to do this is to start with the result and traverse the chain back until we end with cells without precedents.

For each reference in a cell’s formula
  If reference is a literal then
    If function’s parameter labeling insufficient then Transparency = \(-\infty\)
    Else Transparency = Transparency – 1
  Else
    Transparency = Transparency + Reference’s Chain Transparency
  End if
Next

8.6 Model Transparency

A model’s transparency measure is the sum of all occupied cell transparency measures.

For each occupied cell in model
  If cell not a label then Transparency = Transparency + Cell’s Transparency
Next

9 Practical Application

To improve model transparency we must first fix any opaque cells. After that we can consider (as one reviewer noted) if we have inadvertently made our model less transparent by replicating remote source values next to formula cells solely to improve each formula’s transparency score. This just distributes the formula’s transparency score to more cells with each cell worsening our overall model’s transparency score. Our model’s transparency score is an indicator of the total effort required to assess source values. Our goal should be to minimize such effort and thus make models as transparent as practical without increasing effort due to other measures such as complexity or readability.
10 WHAT WAS LEARNED

In developing this metric it became apparent that labeling is a crucial component with requirements varying by value type. Label placement impacts transparency and missing labels can make models opaque.

It also became apparent that some Excel functions can create opaque models giving legitimate reason for standards to bar them.

11 RECOMMENDATIONS

- Label all cell references appropriately.
- Label efficiently to reduce clutter.
- Use freeze panes to keep row and column labels in view (immediate vicinity).
- Seek alternatives to indirect references and if no practical alternatives exist make sure indirect references are constrained to occupied cells.
- When formulas require constants use appropriately named references instead of literals.
- When formulas require remote values use remote references instead of local cells daisy chained to remote cells.
- Use structured references when practical.
- Seek alternatives to error values if practical.
- Fix cells reporting errors not appropriately handled in downstream calculations.
- Favor model transparency over individual cell transparency.

12 SUMMARY

This proposal provides a means by which we can rationally measure model transparency and discern best practices through measurements rather than personal bias. We can also automate these methods to facilitate preparing models for audit.

13 REFERENCES


[JCGM 2008] JCGM. “International vocabulary of metrology — Basic and general concepts and associated terms (VIM)”. 2008. Pg. 2
The Reification of an Incorrect and Inappropriate Spreadsheet Model

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

Once information is loaded into a spreadsheet, it acquires properties that it may not deserve. These properties include believability, correctness, appropriateness, concreteness, integrity, tangibility, objectivity and authority. The information becomes reified. We describe a case study through which we were able to observe at close hand the reification of a demonstrably incorrect and inappropriate spreadsheet model within a small non profit organisation. The organisation continues to plan its future over the next 35 years using a spreadsheet based Discounted Cash Flow model with negative discount rates.

1 INTRODUCTION

[Taleb, 2007] prosaically introduced and summarised the issue which informs the existence and content of this paper:

“… 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”

We first and peripherally encountered the reification of spreadsheet information following the financial crash of 2008 [Croll, 2009]:

“the financial valuations expressed in CDO spreadsheet models were reified in the manner summarised earlier by Taleb”.

The discussion [Myers, 2005] of the study [Thorne, 2012] of the spreadsheet related problems in the UK nuclear fuel industry notes:

“At the centre of the incident are the computer spreadsheets, reified forms of the process of measurement…that was also what enabled the process to be subverted”

There is otherwise a total absence of documented experience of the phenomena of reification of spreadsheet information within the literature. The purpose of this paper is to document a relevant case study which recently became available in the hope that it assists others in their documentation, investigation and management of the phenomenon.

In this paper we introduce the organisation and its historical finances, the financial planning spreadsheet that is the focus of this study, the spreadsheet error issues that were raised and the decision processes they influenced. We provide a summary which includes some learning points which may benefit others in a similar situation. We have anonymised the organisation and its domain of operation, officers and members.


2 LEISURE SERVICES ORGANISATION

2.1 Introduction

This paper is based upon a small non-profit leisure services organisation (LSO). The organisation is a UK limited company and thus its financial accounts are publicly available. It has Community Amateur Sports Club (CASC) status from which it derives some public financial privileges, in particular freedom from taxation on profits. It has been established over 40 years and has been historically grant aided by the UK Sports Council and the UK National lottery. The LSO has historically received other gifts, bequests and donations from a variety of sources. The LSO owns its own property from which it operates (thus it pays no rent) and significant technical assets related to its domain of operation. It provides access to and use of these assets for the benefit of its members (of which there are 50-100) in exchange for annual membership and point of use fees. The domain of operation of the LSO is not relevant and the intention is that the organisation and its directors, officers and members remain anonymous. We have successfully used the cloak of anonymity in previous work [Croll, 2005]. The LSO is managed by a committee and controlled by a smaller number of directors including a chairman and a treasurer who also sit on the committee. All the management are volunteers, are members of the LSO and of some years standing. The management retire by rotation after periods of between three and five years. There has been a plurality of directors, officers, members and reporting accountants and auditors over the years. The management is transparent, with regular meetings and the prompt circulation of detailed minutes, decisions, financial, technical and operational data.

2.2 Key Financial Data

The LSO has been profitable since incorporation and has built up a significant cash and asset base to support its operations. Though initially indebted to a high street bank and one or two benign creditors, the LSO was grant aided in 1996 and received a significant cash bequest in 2001. These enabled the LSO to replace its by then dilapidated technical assets with more up to date assets for members enjoyment. The assets have performed well over the ensuing 20 year period to date (2016) with relatively minor maintenance requirements and have significant remaining life and financial value. The LSO was given another cash sum (a 50% contribution) within the last three years by an anonymous donor in order for it to replace a rented asset with a newer and more efficient asset of its own costing circa £60k plus a further £20k restitution of the rented asset prior to its return.

Table 1 – LSO Key Balance Sheet Data 1993 - 2016

<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>FIXED ASSETS</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Tangible Assets</td>
<td>£156k</td>
<td>£212k</td>
<td>£252k</td>
<td>£247k</td>
</tr>
<tr>
<td>CURRENT ASSETS</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Stock</td>
<td>£3k</td>
<td>£0</td>
<td>£0</td>
<td>£3k</td>
</tr>
<tr>
<td>Debtors</td>
<td>£3k</td>
<td>£8.5k</td>
<td>£5.5k</td>
<td>£8k</td>
</tr>
<tr>
<td>Cash</td>
<td>£3k</td>
<td>£50k</td>
<td>£104k</td>
<td>£115k</td>
</tr>
<tr>
<td>CREDITORS</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>One Year</td>
<td>£-41k</td>
<td>£-20k</td>
<td>£-4k</td>
<td>£-10k</td>
</tr>
<tr>
<td>NET CURRENT</td>
<td>£-35k</td>
<td>£-10k</td>
<td>£105k</td>
<td>£116k</td>
</tr>
<tr>
<td>ASSETS/</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>TOTAL ASSETS</td>
<td>£121k</td>
<td>£201k</td>
<td>£237k</td>
<td>£264k</td>
</tr>
<tr>
<td>CREDITORS</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>More than one</td>
<td>£45k</td>
<td>£0</td>
<td>£0</td>
<td>£0</td>
</tr>
<tr>
<td>CAPITAL AND</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>RESERVES</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Income and</td>
<td>£76k</td>
<td>£201k</td>
<td>£228k</td>
<td>£228k</td>
</tr>
<tr>
<td>Expenditure</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Capital Reserve</td>
<td>£138k</td>
<td>£138k</td>
<td></td>
<td></td>
</tr>
<tr>
<td>SHAREHOLDERS</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>FUNDS*</td>
<td>£76k</td>
<td>£201k</td>
<td>£358k</td>
<td>£364k</td>
</tr>
</tbody>
</table>
We show in Table 1 key balance sheet data (rounded) for the key years of this analysis obtained from the public record [Companies House Beta, 2017] by 24 separate downloads. We show in Figure 1 the key data graphically and in more detail for the 24 year period 1993-2016. It is very likely that none of the members, directors or officers of the LSO have seen the LSO data as depicted in Table 1 and Figure 1, though parts of it will be necessarily familiar to some of them.

The balance sheet data shows the transition from early debt funding to grant aided “equity” funding, modest book profitability over the long term and stable cash, capital and reserves. Trade stocks, creditors and debtors are minimal due to the nature of the organisation. Note that the company is limited by guarantee and has no shareholders. The rest of this analysis focuses upon the years 1996-2016 following acquisition of the main technical assets.

Figure 1 – LSO Key Financial Data

The technical assets, with the exception of the recent acquisition noted above, are fully depreciated with a Net Book Value of effectively zero. All the technical assets have calculated or estimated remaining lives of the order of 20-30 years. The technical assets have an estimated fair market value of £150k and are easily and regularly tradable in a pan-European marketplace. Thus the public accounts significantly understate the assets (and thereby profits) of the company, but are nevertheless properly prepared under the historical cost convention. The near term sale of the technical assets as part of an asset upgrade process would crystallise these profits or they could simply be revalued as happens on an annual basis at other LSO’s of this type.

The LSO’s freehold property was acquired in the late 1980’s and is represented at £195k book value in the accounts and is not depreciated as a matter of policy. Some historical property improvements - £60k - have been capitalised and are included in this figure. The recent profits of the company are also understated because further amounts of property improvement have been booked to the P&L. These uncapitalised discretionary property improvements amount to a further £60k.

We show in Table 2 our summary of the LSO’s estimated actual profits before depreciation 1996-2016.
Table 2 – LSO Estimated Profits Before Depreciation

<table>
<thead>
<tr>
<th></th>
<th>1996-2016</th>
</tr>
</thead>
<tbody>
<tr>
<td>INCOME AND EXPENDITURE</td>
<td>£25,000</td>
</tr>
<tr>
<td>DEPRECIATION</td>
<td>£126,000</td>
</tr>
<tr>
<td>RESIDUAL ASSET VALUE</td>
<td>£150,000</td>
</tr>
<tr>
<td>UNCAPITALISED PROPERTY</td>
<td>£60,000</td>
</tr>
<tr>
<td>TOTAL</td>
<td>£361,000</td>
</tr>
<tr>
<td>PER ANNUM</td>
<td>£18,050</td>
</tr>
</tbody>
</table>

A recent independent review of the LSO’s financial statements by the Development Officer of the LSO’s umbrella organisation confirms that the LSO is “in a good financial position”. It is one of only a handful in the United Kingdom to own its own property and thus pay no rent. Comparison of its financial statements with other LSO’s of the same type and size would reveal that it is one of the wealthiest smaller LSO’s of its type in the United Kingdom.

4 THE FINANCIAL MODEL

4.1 Background

During the last few years members expressed through a series of informal meetings with the directors and a series of resolutions at Annual General Meetings an increasing desire that the directors replace or upgrade the technical assets with more up to date technology. This became imperative as the number of technical assets available for members use decreased recently due to accidental damage, disposal or periods of more extended maintenance. The chairman, directors and management of the LSO were resistant to change and procrastinated at every opportunity. Eventually, the chairman decided to commission a financial review of the LSO which involved the production of an Excel based financial model by two of the committee members, one of whom was a director.

4.1 Structure of the Financial Model

The model was a simple annual cash flow covering the 35 year period from 2016 to 2050 inclusive. The major maintenance and eventual replacement costs for each of the technical assets were tabulated for each year 2016-2050 in 2016 Money Values using a simple mechanism to account for perceived periodicity in major maintenance activities. Thus for the first asset, we have “[refurbish]” estimated at £18k in 4 years followed by £18k every 30 years. So the model would switch in an expenditure of £18k in 2020 and £18k in 2050. Since the first asset has a finite life the second entry was its replacement cost of £50k in 24 years followed (apparently) by its re-replacement of £50k in a further 24 years (i.e. beyond the scope of the model). All the assets ran for their full estimated or calculated lives and were replaced with an asset worth the same as the present (2016) value of the asset. End of life asset values were assumed to be zero. All the other routine operating costs of the technical assets were ignored as these would be covered by point of use revenue which was not included in the model. Major improvement expenditures for the property assets were subject to the same treatment. There were some minor one off expenditures also included in the model and a binary variable for calculating the effect of adding or not adding in the costs of a much desired further technical asset with a net budgeted cost of £40k.
Against cash expenditure as outlined above, the model had the then estimated 2016 opening bank balance of £110k and three choices of future cash income: £4k per annum, £8k per annum and £12k per annum. These were three choices of “profit before depreciation and one offs” based upon the modellers’ view of the company’s historical trading and how this might continue into the future and provide an inbound cash flow. There was no narrative describing how these figures had been arrived at. The formulae within the model show that the figures £12k and £4k were +150% and -50% computations around the central figure of £8k, which figure is at variance with our estimate of £18k pa in Table 2.

The opening bank balance was credited and debited with income and expenditure for the first and then each subsequent year. The carried forward balances were created for each year until 2050 in 2016MV’s. A separate row on the spreadsheet inflated the 2016 carried forward balances at an inflation rate given in the model at 2%. A safety bank balance initially set at £30k was also inflated through the model’s period for comparative purposes.

4.2 LSO Financial Model Output

We show in Figure 2 & 3 the charts depicting the models output which were first presented and discussed at an LSO board meeting in October 2016. Figure 2 shows the model output assuming no further purchases of technical assets beyond end of life replacement and Figure 3 shows model output with an immediate purchase of a new technical asset.

Figure 2 – LSO model output – no new technical asset

Figure 3 – LSO model output – immediate purchase of new technical asset
Figures 2 and 3 suggest an organisation in financial difficulties with medium term insolvency under all three income assumptions whether or not an additional technical asset was purchased. Figures 2 and 3 bear no relation to the history of the organisation obtained from the public record and summarised in Table 1 and Figure 1.

4.3 Results of the LSO Financial Review

A few days after the presentation of the financial model to the LSO board, the results of the financial review were communicated by the chairman to the members of the LSO in late October 2016:

“You will probably be aware that we have been carrying out a detailed review of ... finances recently. The prime reason for this was to confirm what funds .... must reserve for necessary maintenance and, in due course, replacement of .... assets and so allow identification of what amount of the ..... current bank funds may be used to purchase an additional [technical asset]. .....A further reason for the review was to set up a system which will allow the [LSO] to monitor the financial condition of the [LSO] more easily. The findings of that review were presented to the Committee ..... so that we could consider the conclusions to be drawn from it.

The [LSO] is fortunate to own a number of major [technical] assets. Most will have to be replaced at some point in the future; all will need major refurbishments and/or component replacement during their remaining working lives. We have spoken directly to suppliers... [etc, and]... have been able to arrive at a best judgement of when the major expenditure events will occur and what their costs are likely to be.

.....This exercise has shown us that we will have to live within our means if we are to hand over to successive generations .... a financially healthy [LSO]. It is the unanimous conclusion of the Committee that we should not, at this juncture, purchase another [technical asset]. We believe that, unless or until, we can significantly improve our profitability, we should maintain the [technical assets as they now are].....

We intend to further refine the mechanisms used for the review and to continue to use them to update and monitor our financial status. The objectivity they provide will improve the ability of the Committee and Directors to take informed decisions in the best interest of the [LSO].

If you would [like] more information on the financial modelling ......”

Thus the information in the LSO’s spreadsheet based financial model became reified. It was invested with confidence, correctness, authority, integrity, believability, appropriateness and even objectivity. It was set to become the mechanism by which the LSO was to be managed for the future. Most importantly, the review denied any material replacement or upgrading of the LSO technical assets despite a healthy bank balance (unused for two decades), historically secure finances and the members reasonable wishes formally and informally expressed over a prior period of several years.

4.4 Initial challenge to Financial Model

In the initial absence of the financial model itself and the data presented in Table 1, a challenge to the conclusions derived from the financial model was made based upon the
information published in the 2015 statutory accounts which were the latest available at the time:

"....we have conducted a detailed examination of the [LSO]’s financial situation by downloading the [2015] statutory accounts of [LSO] from Companies House.....

These accounts show that the current assets of the [LSO] - principally cash - have increased this last accounting year by £16k from £80k to £96k. This is largely due to the profitable activity of the [LSO] as reflected by the increase in its income and expenditure account. The Net Assets of the [LSO] are £352k....

.....The [LSO] has no debt and is almost free of taxation due to its [CASC] status...

..In summary the statutory accounts of [LSO] show that the [LSO] is profitable, free of debt, has no taxation... and has reasonably provided for the preservation of its assets through an appropriate charge for depreciation. It has net assets in excess of £350k and cash assets in excess of £95k as of its last statutory accounts. These assets have significantly increased since 2012....

[we are] curious about the spreadsheet based financial modelling which has taken place to demonstrate that we cannot afford further [technical assets] that might better match our needs or aspirations......

The two modellers requested a meeting the following evening to discuss the financial model and the above review. Although recollection of the meeting is not exact, it is certain that the quality of the model and the data upon which it relied were positively asserted with undue confidence [Panko, 2003]. No mention was made of any testing [Pryor, 2004; Panko, 2000 & 2006] that had taken place and no documentation [Pryor, 2006] was produced. A copy of the financial model and the presentation made to the board was however promptly provided the next day.

4.5 LSO Financial Model Review

A review of the LSO Financial Model occurred in two phases, much as described in our earlier work [Croll, 2003]. The low level review was kindly performed by a third party not involved with the LSO.

4.5.1 Low level review

The low level review of the LSO financial model revealed no major errors, save for the observation that about half of the model was not in use. There were many examples of poor practice such as embedded constants, complex formulae, strange formatting and a failure to colour code inputs and outputs. There was no evidence of any testing or documentation save a small revision history and a few cell notes.

4.5.2 Initial High Level Review

An initial high level review of the model was rapidly performed and sent directly to the modellers and relevant officers of the LSO. Despite the relatively small size of the model there were five main areas of concern:

- The use of an exponentially inflated cash flow over an extended future period of 35 years for a small organisation which gave the misleading impression of
The evisceration of cash. This point was badly worded in the document sent to the modellers.

- The use of an understated inflation rate of 2% instead of average inflation over the last 20 years, which is nearer to 3% [Bank of England, 2017] which would have “highlighted the absurdity of the model”

- Failure to use the established methodology of Discounted Cash Flow whereby future cash flows are discounted back to the present using an appropriate discount rate in order to facilitate the comparison of investment alternatives. A brief explanation of the DCF methodology was included.

- Use of data derived from overly conservative depreciation rates which greatly understated the gross margin available for the replacement of capital assets.

- No allowance for management’s ability to postpone, reduce, avert or obviate key capital actions through the use of common sense, technology or asset replacement.

The high level review pointed out that the model reified or made concrete potential actions that are in the distant future.

4.5.3 Model test script

Following investigation of the model at high and low level, a very simple test script was devised to demonstrate the models failings. The test script contained the exact keystroke by keystroke detail required to perform the following three tests:

- Use the model to calculate how much cash is required in 2016 in order for the LSO to meet the specified cash safety margin in 2050. This was £442k – four times the 2015 cash balance.

- Show how by changing the inflation rate assumption from 2% to 3% that the amount of cash required in 2016 to meet the 2050 safety margin increases to £639k – six times the 2015 cash balance.

- Show how, by changing the refurbish period for asset one from 30 years (occurs in 2050) to 31 years (would occur in 2051 but outside scope of model) reduces the cash required in 2016 by 35k, demonstrating “edge effects” in the model.

The test script also asked users to:

- Note that the model does not include or provide for cash (and profit) that can be generated by the sale of current fully depreciated assets. i.e. the estimated £150k value of the existing assets is not included in the model.

The test script was sent to the Chairman and Treasurer in January 2017 with a request that the model not be used as a basis for further decision making unless and until the tests had been performed. Unfortunately the model had already delivered its results via the Chairman to the members two months previously as described in section 4.3.
4.6 Model re-implementation

In order to confirm that the LSO financial model was free of material formula error, we re-implimented it using the original maintenance data provided. We were able to cut out the half of the model that was not used, eliminate the complex formulae used to switch in the periodic expenses and perform simple tests around discount rates and asset replacement strategies.

4.7 Further review of the model

During the course of the preparation of this paper the model was subjected to further examination during which the following additional observations were made:

- The model did not relate the easily available historical cash flow depicted in Table 1 and Figure 1 to the projected future cash flows of Figures 2 and 3.
- The model implemented a single “use till death of asset” strategy only. There was no investigation of the value or otherwise of immediately upgrading technical assets by selling them for cash and a book profit and then replacing them with newer and upgraded technology using the otherwise unused cash available. There are many ways of interpreting reality and implementing a spreadsheet model [Banks & Monday, 2002].

4.8 Response to the High level reviews & the Test Script

The chairman responded:

“…..Your fundamental objection to the failure of the model to use the method of discounted cash flow is correct….. I have used DCF for evaluation of projects and the interest rates that you mention are not distant from those that I have come across being proposed by funding agencies. However… to this extent the model is flawed but it does allow attention to be focussed on how best to operate financially and to give some indication of whether corrective action is required…. A sensitivity analysis is usually carried out to show robustness of predictions and this has not been done”

The principal modeller responded:

The model uses a central forecast of £8k for profit from core operations (before .... one-offs and depreciation). The draft 2015-16 financial accounts show a higher number. What's the right long term number? The model is sensitive to this number.

And then:

“Re cash flow, lets ignore the discount/inflation rate for the moment and just focus on the cash flows. The [Asset one] £50k in 2040 should be the net cash outflow of buying a replacement after selling the near end of life [Asset one], otherwise it can
be reduced by whatever we think the market price of [Asset one] will be after 25 yrs further use”.

The LSO director involved in the modelling responded to an explanation of Table 2 as follows:

“We have tried to quantify what we can reasonably see coming down the track and to fund it we need cash, rather than theoretical profit or liquidating current assets, to buy things. We’ve used inflating costs and inflating income and, thereby, inflating profit. We’ve taken informed guesses at the relevant values which we intend to modify as better data becomes available”

To which the reply:

“….when the [LSO] decides to sell [assets] to get something else it will be a statutory requirement to book the sale value as a profit because the ..... assets are fully depreciated. This is not a theoretical profit. It is an actual profit. To financially report otherwise would be unlawful.

“...Assets are routinely sold to support the purchase of other assets. All the [LSO’s] assets are capable of being sold at some time or other and where these amounts are significant - to the tune of £150k - it would be negligent not to include them as part of a competent financial model designed to support the management of the [LSO]”

received no response. It is likely that the director is not aware of the basic tenets of Generally Accepted Accounting Principles (GAAP) or the theory and practice of DCF.

An ex officio member of the committee responded:

“T...think it's reasonable to conclude that our model is not accurate, and basing decisions on a flawed model would be foolish”

There was no response from the chairman and the treasurer to the test script and no evidence that it had been executed.

5 ALTERNATIVE FINANCIAL MODELS

We briefly investigated two alternative financial models which could bring clarity to the decisions faced by the LSO.

5.1 Discounted Cash Flow

This was identified as the preferred capital appraisal evaluation methodology in the high level review. In this application DCF has its own problems, not least the selection of an appropriate discount rate [Croll, 2010]. Also, for positive discount rates, DCF will force significant capital expenditures into the far future supporting the “use till death” strategy of the LSO financial model.

The financial model described in this paper was constructed such that simply changing the inflation rate to a negative number turned the model into a DCF model. The modellers did not appear to realise this. The UK government has traditionally used a discount rate of 6% nominal on publicly funded infrastructure projects. Private firms would use a higher
rate on the same projects. We show below the output of the LSO model with a 6% discount rate (entered as a -6% inflation rate) and a revised £18k inbound annual free cash flow per Table 2.

Figure 4 – LSO model output – immediate purchase of new technical asset using 6% discount rate with revised £18k pa inbound future cash flow

The DCF methodology using the same financial model, identical maintenance data, 6% nominal discount rate and a revised £18k annual inbound cash flow shows the organisation to be in good prospective health for the future. The projected future cash balances are similar to the existing and included the immediate purchase of a new technical asset, plus all the maintenance and improvements in the Financial Model.

5.2 Return on Capital Employed (ROCE)

The LSO Shareholders Funds for 1996 in Table 1 shows a balance of £201k. This increased to £364k in 2016. We might reasonably add to this 100% of the uncapitalised property improvement expenditure of Table 2 (£60k) and the fair market value of the fully depreciated technical assets of Table 2 (£150k). This suggests an adjusted Shareholders Funds account of £574k as at 2016. Thus the LSO achieved an adjusted ROCE of £574k / £201k = 285% over the 20 year period from 1996 or 5.4% compound per annum nominal, 2.8% real.

The LSO overall ROCE of +2.8% real contrasts markedly with the -2.8% per annum real that the LSO has achieved for its £133k cash balance over the period since 2001. The cumulative deflationary loss of capital that the LSO has suffered over the last 15 years due to its large cash holding of benefactors funds is approximately £69k in 2016 MV’s, less a small amount of interest.

The ROCE methodology suggests that the LSO should immediately employ most of its available cash as technical assets in its own business save any cash contingency requirements.

The ROCE of 5.4% nominal supports a discount rate for use in the financial modelling of 6% nominal.
6 CONCLUSION

We have described a simple spreadsheet that was used in a financial review of a small Leisure Services Organisation [LSO] in order to evaluate the wisdom or otherwise of the upgrade, replacement or enhancement of its technical assets.

The spreadsheet presented to the management of the organisation suggested that the organisation was about to enter serious financial difficulties and it could not afford to upgrade or extend its assets. These projected circumstances were at variance with the solid historical financial performance of the organisation over the previous 20 or more years.

We have shown how the model became reified. That is to say the spreadsheet was invested with a variety of attributes including believability, correctness, appropriateness, concreteness, integrity, tangibility, objectivity and authority. None of these attributes were deserved as we were able to subsequently identify and describe numerous significant weaknesses and strategic flaws in the model. We revealed these issues through a straightforward software engineering process involving the performance of a low level model review, a high level model review, the production of a simple test script and the re-engineering of the model.

We investigated the historical circumstances of the LSO by downloading its publicly available accounts and transaction history for the prior 20 years and re-estimated its actual historical profitability. We show that the LSO’s future projected free cash flow has been underestimated by a factor of more than two.

We then used the original model with a positive discount rate of 6% (rather than a negative discount rate of 2%) and our revised future free cash flow to perform a DCF analysis which showed that the LSO was easily capable of affording the new technical and other assets that its members aspired to.

It appears to be the case that the organisation has been planning (and continues to plan) its future over a 35 year period using a spreadsheet based Discounted Cash Flow Model with negative discount rates (contrary to centuries of tradition). It continues to use a future inbound cash flow underestimated by a factor of more than two.

We used a ROCE analysis to show that the LSO had generated a 5.4% nominal compound rate of return on its capital over the prior 20 years. This implied that the significant cash that it has had languishing in the bank would be far better employed for the future in the assets and business of the LSO.

As of the time of writing in May 2017 despite a change in leadership, no changes have been made to the technical assets and the LSO spreadsheet is still extant leaving one exasperated LSO member (a former LSO treasurer) to write:

“The [LSO] and the wonderful facilities were built by adventurous forward looking people but the doom and gloom spreadsheet has permeated the soul of the [LSO] and has worked its insidious magic.”
The reification of the deeply flawed spreadsheet that we have examined has had the unnecessary and negative consequence of stagnating the organisation’s future development. Our experience informs us that deeply flawed but reified spreadsheets have inflicted serious damage on other organisations too.

7 RECOMMENDATIONS

We significantly underestimated the amount of resource that would be required to challenge the simple spreadsheet model at the heart of this investigation. Those who follow, who might be working on spreadsheets a hundred times more complex, might wish to note the following:

- Prior to challenging an existing model, you will need to perform a full and effective high and low level model review and possibly develop a series of model test scripts. Any flaws in the review process or the test scripts will undermine the challenge.

- You may need to acquire or reacquire and then validate the relevant model source data.

- If the target model is strategically, tactically or computationally incorrect you may need to build another model in order to figure out what the right answer is in the first place. You will then need to perform a thorough high and low level model review on the challenger model. And then compare and contrast the differences between the challenger model and the existing model.

- Do not expect the management of the organisation you are dealing with to have the capacity or interest to comprehend the errors, limitations or flaws observed by you in their financial modelling or accounting methods.

Whilst the above processes are taking place – all at the same time, with great time urgency and in a seemingly haphazard event driven order - the original model (flawed in your opinion) at the heart of your challenge will be seeping deeper and deeper into the organisation.

Spreadsheets are ubiquitous and error prone but the scientific literature gives good guidance as to the processes required to investigate and correct the errors. It would be wise to follow the literature. We estimate that the time required to challenge a reified model is at least three to six times the time required to create the (probably untested) target model in the first place. This can be extraordinarily expensive and it may be cheaper and far less stressful to simply walk away and seek a fresh challenge elsewhere.

ACKNOWLEDGEMENTS

We thank Patrick O’Beirne for performing the low level review at short notice. We thank an Auditor, a Software Engineer, a City of London Banker, a Fellow of the Irish Computer Society and a Professor of Law for their promptly provided informal reviews of this paper. We thank the anonymous referees for their helpful comments.
REFERENCES


Structured Spreadsheet Modelling and Implementation with Multiple Dimensions - Part 1: Modelling

Paul Mireault
Founder, SSMI International
Honorary Professor, HEC Montréal
Paul.Mireault@SSMI.International

ABSTRACT

Dimensions are an integral part of many models we use every day. Without thinking about it, we frequently use the time dimension: many financial and accounting spreadsheets have columns representing months or years. Representing a second dimension is often done by repeating blocks of formulas in a worksheet of creating multiple worksheets with the same structure.

1 Introduction

Most organizations deal with dimensions, without calling them as such. For example:

- Products, product categories, product types.
- Clients, client types, clientstatus.
- Markets or sectors, like education or health.
- Locations. They may be geographical (such as countries, continents, regions) or specific (such as manufacturing plants, warehouses) or arbitrary (such as sales regions).

In this paper, we will first present some examples of multidimensional spreadsheets. Then, we will do a brief summary of the conceptual modelling methodology we use to represent the problem we wish to solve with a spreadsheet. We will then present basic concepts of dimensions, variables and multidimensional expressions. We conclude with a case study and describe its complete multidimensional model.

2 Examples of multidimensional spreadsheets

Microsoft Excel has a tool called Pivot Table that can help the spreadsheet developer present a multidimensional dataset in a two-dimensional table, using rows and columns to represent more than one dimension. While Pivot Tables are good for presenting data, they are less suited for presenting business spreadsheets. The principal reason is that Pivot Tables require that their source is organized vertically as tables: each column represents a variable, and the rows represent the repeated values. The spreadsheets we are interested in are organized horizontally: the rows represent variables and the columns are the repeated values. We could transpose an horizontal structure into a vertical one, but this extra step does not alleviate Pivot Table’s other shortcoming. The major reason we feel that Pivot Tables are not suitable for spreadsheet that represent a model used to analyse scenarios, as opposed to a spreadsheet containing data, is that they do not update their results when their base data changes. For that reason, the results produced by Pivot Tables cannot be used in the calculations of other variables.

Even though a spreadsheet has two dimensions, rows and columns, it usually represents only one dimension. Most business spreadsheet use rows for variables, leaving the columns for one dimension, like the Time dimension.

One approach is to create one worksheet for each instance of a dimension and implementing the other dimensions inside those worksheets. For example, if the dimension is Region, there could be four
worksheets for North, East, West and South. One could then build a fifth worksheet with consolidating formulas.

This method is proposed by (Sartain, 2014) where the author describes using 13 worksheets, one for each month and one for consolidation, and each worksheet assigns expense variables, in rows, to different persons, in columns (see Figure 1). The author also describes the maintenance task of adding or removing an account, which involves performing the same operation 13 times, once for each worksheet. She also strongly suggests, when adding an account, to add the row somewhere in the middle of the other accounts to make sure that the subtotal row includes it in its calculation.

(Brandewinder, 2008) has a spreadsheet with three dimensions: Quarter, Product and Region (see Figure 2). The Product dimension is presented as different worksheets, the Quarter dimension as columns and the Region dimension as blocks of repeated formulas.

One can use an entire worksheet to represent one two-dimensional variable. Figure 3 shows an unpublished example where the two-dimensional variable **Border Right Indic** is implemented in its own worksheet.
Figure 3 Example of a worksheet used for one two-dimensional variable

(Savage, 1997) describes two important problems with using dimensions in spreadsheets. First is *scalability*, which involves changing the cardinality of a dimension. He concludes that spreadsheets rarely scale well. Second is *hyper-scalability*, which involves changing the dimensions themselves, such as adding more dimensions. His conclusion is, succinctly, “Forget it”.

Multi-dimensional spreadsheets have also been used in specific optimization problems. (Kumar, 2014) describes a course scheduling problem with three dimensions: faculty, course and timeslot. A textbook by (Powell & Baker, 2013) presents many classic Management Science problems such as the Network Flow, the Assignment and the Traveling Salesman. While they present some multi-dimensional problems, their spreadsheets are specific to each problem.

3 The Conceptual Model

In Information Systems development, the stage were the requirements are specified produces the *conceptual model*. The conceptual model describes what the system must do, with little reference to the technology that will be used for the implementation.

(Grossman & Özlük, 2010) in their study of three spreadsheet engineering methodologies found that two of them do not discuss modeling and the other requires a detailed output specification.

Other researchers described building a conceptual model before implementing the spreadsheet, even though they did not call it *conceptual modelling*. The Jackson Structured Diagram, a diagraming technique based on programming concepts, has been proposed by (Knight, Chadwick, & Rajalingham, 2000). Their diagram has some similarities with the simple Formula Diagram of (Mireault, 2017), but they do not show how to extend it to a one dimension model. (Powell & Baker, 2013) use Influence Charts to model a problem and give general advice on how to implement it in a spreadsheet. While their examples show a one-dimension spreadsheet, with Quarters, their Influence Chart does not show which variables belong to the Quarter dimension.

3.1 The Formula Diagram of the SSMI Methodology

(Mireault, 2017) presents a methodology for developing spreadsheets based, primarily, on following the process used in information systems development, where the requirement specifications is separate from the implementation. The Structured Spreadsheet Modelling and Implementation (SSMI) methodology consist of building a conceptual model of the spreadsheet’s variables and their formulas before doing the implementation. The conceptual model is composed of a Formula Diagram.
(Figure 4) and a Formula List (Figure 5), and they are used later to do the implementation of the spreadsheet.

![Diagram of Formula Diagram](image)

Figure 4 Example of a Formula Diagram, taken from (Mireault, 2017)

The Formula Diagram uses the following symbols:

- Triangles and squares represent data values. The squares are used for input values, data that the developer wants to implement in an Interface worksheet to allow the user to easily modify its value. The other data, the triangles, represent data that don’t change often and will be implemented in their own specific data worksheets.

- Circles and ovals represent calculated variables. The ovals are used for results that the developer wants to display in the Interface worksheet, close to the input data so the user can quickly see the impact of changing an input value. The circles are variables of less interest to the user and are implemented in their own specific model worksheets.

- Arrows indicate which variables are involved in the calculation of the variable receiving them.

- The dash-bordered box represents a dimension, also called entity. All the variables appearing within the box have multiple values, one value for each instance of the repeated entity. For example, if we have three regions, then the variable **Regional Demand** has three values. All the variables appearing outside the dashed box have a single value.
While the Formula Diagram gives a global view of the model, the corresponding Formula List gives a detailed view, with all the formulas written in an Excel-like form, using variable names.

The Formula Diagram is inspired from the Influence Diagram, as presented in (Bodily, 1985). The Influence Diagram has a richer set of modeling concepts, such as uncertainty in the values of data variables and uncertainty in the formulas of calculated variables. But the Influence Diagram has no representation of groups of repeating variables, which the Formula Diagram represents with a dash-bordered box.

4 Multidimensional modelling concepts

At this point, we invite the reader to read the case study presented in the appendix so that they can get a better appreciation of the concepts we present in this section.

4.1 Dimensions

A dimension is a set of values that serve to characterize a specific value. The set of values form a partition. A partition, in set theory, represent subsets whose intersections, taken two by two, are null, and whose union is the universal set, that is the set of all values. In plain language, it means that there is no overlap and all possibilities are covered.

For example, if we use the dimension Region to characterize clients and we have the set of values \{Mountain, Valley, Lake\}, a client must belong to one of the regions (all possibilities are covered) and cannot belong to two regions (no overlap).

4.2 Dimension sets

A dimension set is a set comprised of 0 or more dimension, and a variable belongs to a specific dimension set. Often, the variable name we use gives a clue to the dimension set it belongs to: the variable named Monthly Production belongs to the dimension set (Month) and the variable Monthly Regional Sales belongs to the dimension set (Month, Region).

We will say that a variable belonging to the empty, (), dimension set is dimensionless. We will also say that dimension sets composed of only one dimension are basic. Finally, the dimension set composed of all the dimensions is called the full dimension set.
If we have \( n \) dimensions, then we have \( 2^n \) possible dimension sets, ranging from the empty set to the set of all dimensions. Thus, when we have only one dimension, like Time, a variable either belongs to the (Time) dimension set or is dimensionless. If we have two dimensions, like Month and Region, a variable either belongs to the (Month, Region) dimension set, the (Month) dimension set, the (Region) dimension set or the () dimension set.

### 4.3 Defining variables

In usual mathematical notation, a variable’s dimension set is indicated by subscripts. Thus, the two variables described above would be written like this: \( \text{Unit Production Cost}_{\text{Product}} \) and \( \text{Unit Delivery Cost}_{\text{Region}} \). It is redundant to specify the dimension set in the variable’s name and in the subscripts: we will only do so in this section because we want to make sure that the dimensions are clear.

There are mathematical rules to remember when dealing with expressions involving variables of different dimension sets. We usually apply them without thinking about it because they are common sense. We will describe the rules and show how they are represented in a Formula Diagram and a Formula List.

**Rule 1:** The dimension set of a formula is the union of the dimension sets of all the variables that are part of its definition.

**Example 1:**

- \( \text{Unit Production Cost} \) is of dimension set \((\text{Product})\).
- \( \text{Unit Delivery Cost} \) is of dimension set \((\text{Region})\).
- \( \text{Unit Cost} = \text{Unit Production Cost} + \text{Unit Delivery Cost} \) is thus of dimension set \((\text{Product}, \text{Region})\).
- The mathematical representation of the formula is:

\[
\text{Unit Cost}_{\text{Product, Region}} = \text{Unit Production Cost}_{\text{Product}} + \text{Unit Delivery Cost}_{\text{Region}}
\]

- Figure 6 illustrates how this variable definition is shown in a Formula Diagram.

![](image)

*Figure 6 Defining a two-dimensional variable from two one-dimensional variables*

**Example 2:**

- \( \text{Annual Sector-Product Unit Sales} \) is of dimension set \((\text{Sector, Product})\).
- \( \text{Monthly Sales Distribution per Sector} \) is of dimension set \((\text{Month, Sector})\).
- \( \text{Monthly-Sector-Product Unit Sales} = \text{Annual Sector-Product Unit Sales} * \text{Monthly Sales Distribution per Sector} \) is thus of dimension set \((\text{Month, Sector, Product})\).
- The mathematical representation of the formula is:

\[
\text{Monthly-Sector-Product Unit Sales}_{\text{Month, Sector, Product}} = \text{Annual Sector-Product Unit Sales}_{\text{Sector, Product}} * \text{Monthly Sales Distribution per Sector}_{\text{Month, Sector}}
\]

- This is shown in Figure 7.
Rule 2: Besides aggregation, a variable can only be defined with variables having a dimension set that is a subset of its own.

Example:

- **Monthly-Sector-Product Unit Sales** is of dimension set (Month, Sector, Product)
- It can be defined with variables of dimension sets (Month, Sector, Product), (Month, Sector), (Month, Product), (Sector, Product), (Month), (Sector), (Product) and ()
- It cannot be defined with variables of dimension sets (Month, Region) or (Product, Region)
- Figure 6 and Figure 7 are also illustrations of this rule.

Rule 3: In the case of an aggregation, a variable can only be defined with a variable having a dimension set that is a superset of its own.

Example 1:

- **Regional Unit Sales** is of dimension (Region).
- It can be aggregated from a variable of dimension sets (Month, Product, Region), (Sector, Region) or (Month, Region).
- It cannot be aggregated from a variable of dimension sets (Month, Sector) or (Product).
- In the Formula List, we would write it as =SUM(Monthly-Product-Region Unit Sales) or =SUM(Monthly-Region Unit Sales). Mathematically, both formulas are equivalent.
- The mathematical representations of the two formulas are:

\[
\text{Regional Unit Sales}_{\text{Region}} = \sum_{\text{Month, Product, Region}} \text{Monthly-Product-Region Unit Sales}
\]

or

\[
\text{Regional Unit Sales}_{\text{Region}} = \sum_{\text{Month, Region}} \text{Monthly-Region Unit Sales}
\]
• Figure 8 shows how we would present the two formulas in the Formula Diagram.

![Figure 8 Two ways to define the same aggregate variable](image)

Example 2:
• **Regional-Product Unit Sales** is of dimension set \((Product, Region)\).
• It can be aggregated from a variable of dimension sets \((Month, Product, Region)\), \((Sector, Product, Region)\) or \((Month, Sector, Product, Region)\).
• The mathematical representation of the formula is:

\[
\text{Regional-Product Unit Sales}_{Product, Region} = \sum_{Month} \text{Monthly-Product-Region Unit Sales}_{Month, Product, Region}
\]

• It cannot be aggregated from a variable of dimension sets \((Month, Region)\) or \((Product)\).

Example 3:
• **Total Unit Sales** is of dimension set \((\_\_, \_\_, \_\_, \_\_)\).
• It can be aggregated from a variable of any dimension set.
• In the Formula List, we would write it as \(=\text{SUM}(\text{Monthly-Sector-Product-Region Unit Sales})\) or \(=\text{SUM}(\text{Monthly-Product-Region Unit Sales})\). Mathematically, both formulas are equivalent.
• The mathematical representations of the two formulas are:

\[
\text{Total Unit Sales} = \sum_{Month \text{ Sector Product Region}} \text{Monthly-Sector-Product-Region Unit Sales}_{Month \text{ Sector Product Region}} 
\]

or

\[
\text{Total Unit Sales} = \sum_{Month \text{ Product Region}} \text{Monthly-Product-Region Unit Sales}_{Month \text{ Product Region}} 
\]

In the Formula Diagram, the dimensionless variable is shown outside of any box, as illustrated in Figure 9.
5 Conclusion

In this paper, we extended the one-dimension conceptual model of (Mireault, 2017) to model a multidimensional problem. By building the conceptual model before doing the implementation, the spreadsheet developer is not trying to solve two different problems at the same time: *How do I calculate this variable?* and *How do I implement this in my spreadsheet?*

An experiment by (O'Donnel, 2001) showed that using a diagramming technique, an Influence Diagram in this case, did not take significantly more time to produce the final spreadsheet and those spreadsheets had significantly less errors of the type “omitted factors” than those of the control group. The problem submitted to the test subjects was a relatively simple one, with one time dimension of two periods. It would be interesting to reproduce the experiment with a more complex multidimensional problem such as the one presented in the Appendix.

Part 2 of this paper will present a structured methodology to implement the multidimensional model.
6 Appendix – Case Study

In this section, we present a pedagogical case study to illustrate the concepts presented in this paper. The solution is not unique: there are many ways of calculating some variables, as illustrated above.

6.1 The Acme TechnoWidget Company

The Acme TechnoWidget Company produces and sells widgets. It produces two products (a Standard widget and a Deluxe widget) and its salesforce is assigned to four major sectors: Government, military, education, and private.

Market research has established that the annual demand for widgets depends on each sector’s Standard widget price. The Pricing Director explains:

*We start by setting a global base price. Then, for each sector, we tell our salesforce that they can offer a rebate. For instance, we offer a 70% rebate to the education sector and it’s 10% for the private sector because purchases are usually made by researchers with limited funds. The military sector gets a 20% rebate and the government 40%. This is not made public: all our price lists show the base price, but our clients in each sector are aware of the rebate they can get.*

Each sector reacts differently to a change of price. We consulted with a market research expert and she came up with multiple demand functions, one for each sector. The demand function estimates a sector’s annual demand for a given base price. The demand function has the form \( \frac{B}{\text{Price}^A} \). The parameters \( A \) and \( B \) are different for each sector, and \( \text{Price} \) is the sector’s price, after the rebate. This table shows the values the expert gave us:

<table>
<thead>
<tr>
<th>Sector</th>
<th>Government</th>
<th>Military</th>
<th>Private</th>
<th>Education</th>
</tr>
</thead>
<tbody>
<tr>
<td>Rebate Percentage</td>
<td>40%</td>
<td>20%</td>
<td>10%</td>
<td>70%</td>
</tr>
<tr>
<td>DemParA</td>
<td>3.593437587</td>
<td>3.46315031</td>
<td>3.187228762</td>
<td>4.114496316</td>
</tr>
<tr>
<td>DemParB</td>
<td>22858963442</td>
<td>22858963442</td>
<td>22858963442</td>
<td>22858963442</td>
</tr>
</tbody>
</table>

The price of the Deluxe widget is 45% higher than the Standard widget.

The Sales Manager explains the sales pattern:

*The annual demand of each Sector is split between the Standard and Deluxe product types, but the distribution is very different in each sector. For instance, in the education sector, with its limited funds, the split is 80%-20% and it is 25%-75% in the military sector. I guess these guys always go for the best, and they have higher budgets. The distribution is 65%-35% for the government sector and 40%-60% for the private sector. The ratios are then applied to the sector’s annual demand to get the annual demand by product.*

Another interesting pattern is the distribution of sales during the year. We noticed that our clients buy more just before the end of their fiscal year, when some want to spend their budget surpluses, and the beginning, when others have new funds allotted. Each sector has a different pattern, and we noticed that it is pretty stable year after year.

<table>
<thead>
<tr>
<th></th>
<th>Government</th>
<th>Military</th>
<th>Private</th>
<th>Education</th>
</tr>
</thead>
<tbody>
<tr>
<td>Jan</td>
<td>9%</td>
<td>8%</td>
<td>12%</td>
<td>6%</td>
</tr>
<tr>
<td>Feb</td>
<td>10%</td>
<td>9%</td>
<td>11%</td>
<td>8%</td>
</tr>
<tr>
<td>Mar</td>
<td>12%</td>
<td>10%</td>
<td>9%</td>
<td>9%</td>
</tr>
</tbody>
</table>
Sales to a sector are not uniformly distributed by region. For example, there are more universities in the South-West than in the West. The following table shows the distribution of a sector’s sales by region. With it, we can calculate the expected monthly sales per product per region, which helps our Logistics Department do its planning.

<table>
<thead>
<tr>
<th>Month</th>
<th>Region</th>
<th>Government</th>
<th>Military</th>
<th>Private</th>
<th>Education</th>
</tr>
</thead>
<tbody>
<tr>
<td>Apr</td>
<td>N</td>
<td>25%</td>
<td>52%</td>
<td>22%</td>
<td>24%</td>
</tr>
<tr>
<td>May</td>
<td>SE</td>
<td>18%</td>
<td>13%</td>
<td>21%</td>
<td>15%</td>
</tr>
<tr>
<td>Jun</td>
<td>SW</td>
<td>18%</td>
<td>18%</td>
<td>17%</td>
<td>32%</td>
</tr>
<tr>
<td>Jul</td>
<td>E</td>
<td>22%</td>
<td>0%</td>
<td>25%</td>
<td>17%</td>
</tr>
<tr>
<td>Aug</td>
<td>W</td>
<td>17%</td>
<td>17%</td>
<td>15%</td>
<td>12%</td>
</tr>
<tr>
<td>Sep</td>
<td>Total</td>
<td>100%</td>
<td>100%</td>
<td>100%</td>
<td>100%</td>
</tr>
</tbody>
</table>

The costs of producing a widget are $48 and $72 for the Standard and the Deluxe widget respectively. The monthly fixed costs for this year are $20000. Delivery costs depend solely on the region and are shown in this table:

<table>
<thead>
<tr>
<th>Region</th>
<th>North</th>
<th>South-East</th>
<th>South-West</th>
<th>East</th>
<th>West</th>
</tr>
</thead>
<tbody>
<tr>
<td>Unit Delivery Cost</td>
<td>$10.25</td>
<td>$9.73</td>
<td>$9.58</td>
<td>$8.26</td>
<td>$11.02</td>
</tr>
</tbody>
</table>

The company CEO wants to see the following results:

- The monthly sales amount and units per product.
- The monthly units, sales amount, costs and profit.
- The total profit.
6.2 Acme TechnoWidget Company Formula Diagram

6.3 Acme TechnoWidget Company Formula List

<table>
<thead>
<tr>
<th>No</th>
<th>Variable</th>
<th>Type</th>
<th>Dimension Set</th>
<th>Value / Formula</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Base Price</td>
<td>Input</td>
<td></td>
<td>100</td>
</tr>
<tr>
<td>2</td>
<td>Base Price Multiplier</td>
<td>Data</td>
<td>Product</td>
<td>(1, 1.45)</td>
</tr>
<tr>
<td>3</td>
<td>Unit Production Cost</td>
<td>Data</td>
<td>Product</td>
<td>list of values</td>
</tr>
<tr>
<td>4</td>
<td>Rebate Percentage</td>
<td>Data</td>
<td>Sector</td>
<td>list of values</td>
</tr>
<tr>
<td>5</td>
<td>Sector Price Factor</td>
<td>Calculated</td>
<td>Sector</td>
<td>1-Rebate Percentage</td>
</tr>
<tr>
<td>6</td>
<td>Sector Base Price</td>
<td>Calculated</td>
<td>Sector</td>
<td>Base Price * Sector Price Factor</td>
</tr>
<tr>
<td>7</td>
<td>DemParA</td>
<td>Data</td>
<td>Sector</td>
<td>list of values</td>
</tr>
<tr>
<td>8</td>
<td>DemParB</td>
<td>Data</td>
<td>Sector</td>
<td>list of values</td>
</tr>
<tr>
<td>9</td>
<td>Sector Annual Demand Units</td>
<td>Calculated</td>
<td>Sector</td>
<td>DemParA*DemParB^-Sector Base Price</td>
</tr>
<tr>
<td>10</td>
<td>Unit Delivery Cost</td>
<td>Data</td>
<td>Region</td>
<td>list of values</td>
</tr>
<tr>
<td>11</td>
<td>PR Unit Cost</td>
<td>Calculated</td>
<td>Product-Region</td>
<td>Unit Production Cost + Unit Delivery Cost</td>
</tr>
<tr>
<td>12</td>
<td>Product Distribution per Sector</td>
<td>Calculated</td>
<td>Sector-Product</td>
<td>list of values</td>
</tr>
<tr>
<td>13</td>
<td>Annual Sector-Product Unit Sales</td>
<td>Calculated</td>
<td>Sector-Product</td>
<td>Sector Annual Demand Units * Product Distribution per Sector</td>
</tr>
<tr>
<td>14</td>
<td>Price</td>
<td>Calculated</td>
<td>Sector-Product</td>
<td>Sector Base Price * Base Price Multiplier</td>
</tr>
<tr>
<td>15</td>
<td>Annual Sector-Product Unit Sales</td>
<td>Calculated</td>
<td>Sector-Product</td>
<td>Annual Sector-Product Unit Sales * Price</td>
</tr>
<tr>
<td>No</td>
<td>Variable</td>
<td>Type</td>
<td>Dimension Set</td>
<td>Value / Formula</td>
</tr>
<tr>
<td>----</td>
<td>----------------------------------------------</td>
<td>--------------</td>
<td>-------------------------</td>
<td>--------------------------------------------------------------------------------</td>
</tr>
<tr>
<td></td>
<td>Product Sales Amount</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>16</td>
<td>Region Sales Distribution per Sector</td>
<td>Data</td>
<td>Sector-Region</td>
<td>list of values</td>
</tr>
<tr>
<td>17</td>
<td>Monthly Sales Distribution per Sector</td>
<td>Data</td>
<td>Month-Sector</td>
<td>list of values</td>
</tr>
<tr>
<td>18</td>
<td>MSP Unit Sales</td>
<td>Calculated</td>
<td>Month-Sector-Product</td>
<td>Annual Sector-Product Unit Sales * Monthly Sales Distribution per Sector</td>
</tr>
<tr>
<td>19</td>
<td>MSP Sales Amount</td>
<td>Calculated</td>
<td>Month-Sector-Product</td>
<td>Annual Sector-Product Sales Amount * Monthly Sales Distribution per Sector</td>
</tr>
<tr>
<td>20</td>
<td>MSPR Unit Sales</td>
<td>Calculated</td>
<td>Month-Sector-Product-Region</td>
<td>MSP Unit Sales * Region Sales Distribution per Sector</td>
</tr>
<tr>
<td>21</td>
<td>MSPR Variable Cost</td>
<td>Calculated</td>
<td>Month-Sector-Product-Region</td>
<td>MSPR Unit Sales * PR Unit Cost</td>
</tr>
<tr>
<td>22</td>
<td>Monthly Variable Cost</td>
<td>Calculated</td>
<td>Month</td>
<td>SUM(MSPR Variable Cost)</td>
</tr>
<tr>
<td>23</td>
<td>Monthly Unit Sales</td>
<td>Output</td>
<td>Month</td>
<td>SUM(MSPR Unit Sales)</td>
</tr>
<tr>
<td>24</td>
<td>Monthly Sales Amount</td>
<td>Calculated</td>
<td>Month</td>
<td>SUM(MSP Sales Amount)</td>
</tr>
<tr>
<td>25</td>
<td>Monthly Fixed Cost</td>
<td>Data</td>
<td></td>
<td></td>
</tr>
<tr>
<td>26</td>
<td>Monthly Costs</td>
<td>Calculated</td>
<td>Month</td>
<td>Monthly Fixed Cost + Monthly Variable Cost</td>
</tr>
<tr>
<td>27</td>
<td>Monthly Profit</td>
<td>Calculated</td>
<td>Month</td>
<td>Monthly Sales Amount - Monthly Costs</td>
</tr>
<tr>
<td>28</td>
<td>MPR Unit Sales</td>
<td>Output</td>
<td>Month-Product-Region</td>
<td>SUM(MSPR Unit Sales)</td>
</tr>
<tr>
<td>29</td>
<td>MP Unit Sales</td>
<td>Output</td>
<td>Month-Product</td>
<td>SUM(MSP Unit Sales)</td>
</tr>
<tr>
<td>30</td>
<td>MP Sales Amount</td>
<td>Output</td>
<td>Month-Product</td>
<td>SUM(MSP Sales Amount)</td>
</tr>
<tr>
<td>31</td>
<td>Total Profit</td>
<td>Output</td>
<td></td>
<td>SUM(Monthly Profit)</td>
</tr>
</tbody>
</table>
7 References


Mitigating Spreadsheet Model Risk with Python Open Source Infrastructure

Oliver Beavers
Director, Trivium Financial Group
oliver@triviumfinancialgroup.com
969 2nd St. SE, Charlottesville, VA 22902

ABSTRACT

Across an aggregation of EuSpRIG presentation papers, two maxims hold true: spreadsheets models are akin to software, yet spreadsheet developers are not software engineers. As such, the lack of traditional software engineering tools and protocols invites a higher rate of error in the end result. This paper lays groundwork for spreadsheet modelling professionals to develop reproducible audit tools using freely available, open source packages built with the Python programming language, enabling stakeholders to develop clearly defined model “oracles” with which to test and audit spreadsheet calculations against.

1. INTRODUCTION

Increasingly, publications focused on spreadsheet risk – whether from EuSpRIG, or project finance modelling books – have begun to lean towards a modelling methodology that uses combinations of array formulas and array-level range names [Swan, 2016]. Other approaches to spreadsheet modelling such as the FAST standard propose otherwise, favoring an arguably more readable approach with the use of names only as they apply to external links [FAST, 2017].

The human error of spreadsheet modelling has been well established. EuSpRIG’s Horror Stories [EuSpRIG, 2017] provide accounts of significant financial loss due to spreadsheet error. Additional research has further recognized these errors, and makes calls for additional testing, particularly in larger spreadsheet models where the likelihood of at least one per-cell error is significantly higher [Panko, 2015].

Outside of the spreadsheet community, the Python programming language has grown significantly in user adoption in large part due to its easy to use syntax, and the strength of the open source computing libraries built on top of it. One of the more recent innovations for the language is that of the Jupyter Notebook: a highly interactive computing environment with a user-friendliness similar to that of Excel’s.

Though Python’s modelling applications have traditionally focused on quant finance and data science, the user-friendliness of its Jupyter Notebook interface, high quality computing and code-testing packages, and (as of recent) strong integration with Excel, make it a prime candidate for
replicating modules of detailed financial models, which may be developed once, and reused in a fashion.

This paper defines an introductory approach for spreadsheet professionals to learn to utilize Python’s numerical capabilities by creating a simple financial model in MS Excel, and developing the same model in Python which may be used as an oracle to test spreadsheet calculations against. With less effort than likely imagined by those unfamiliar with the language, spreadsheet developers can create a comprehensive library of Python-based tools to test spreadsheet results across numerous disciplines.

2. PREVIOUSLY PRESCRIBED SOLUTIONS

At a single spreadsheet analysis level, Aurigemma and Panko classify spreadsheet error detection protocols across three categories: testing (a tool traditionally used in software engineering), inspection, and spreadsheet static analysis tools (SSATs).

Inspection is a particularly cost and labor-intensive process: it requires teams of employees to inspect code cell-by-cell at multiple intervals in order to prevent cognitive fatigue, and missed results. Even then, this process only yields an error detection result range of 60-80% under the most optimal of circumstances [Aurigemma and Panko, 2010].

SSATs take a different approach. As software – usually in the form of Excel add-ins – SSATs use various tools such as pattern matching and visual mapping in order to point a user to likely problematic cells.

Across numerous tests, SSATs have shown to work in a similar degree of accuracy compared to the more labor-intensive inspection protocols, yet question remains about specifically what errors they perform well in detecting, and furthermore, results across tools lack in consistency. Kulesz and Ostberg describe this lack of unification in results as an issue of different tools, different configurations, different ways of showing output, but most importantly, different hidden software assumptions that provide different results for even the most-simple of patterns such as “constants in formulas” errors. Furthermore, certain SSATs run into trouble with Excel version upgrades; the authors call for tool vendors to decouple the core audit/inspection functions from the front-end (interface-level) execution environment, and they find only two tools which do so [Kulesz and Ostberg, 2013].

A third protocol prescribed by Aurigemma and Panko is the practice of testing where spreadsheets would be tested against calculations provided by an oracle model. Aurigemma and Panko note that while testing may be one of the better ways to confirm the accuracy of a model in question, the Excel environment does not provide much in the way of testing tools; as such, the testing of complex spreadsheets would require rebuilding another complex spreadsheet – an oracle – to test against. Again, this approach proves labor intensive. Given the combinatorial complexity of large, complex spreadsheets with many inputs and calculations performed, oracles developed in order to test complex spreadsheets may be prone to the same or different errors originally conceived, delivering a lack of clarity as to which spreadsheet actually provides the proper results [Aurigemma and Panko, 2010].

Shubbak and Thorne approach the spreadsheet error problem from an organizational level by developing a program which assesses risks of spreadsheets within an organization, in an effort to provide decision-makers or auditors with the most important spreadsheets to place their focus on.
A key issue, the authors note, is that of redundancy, and a lack of centralization. Due to the ease of developing spreadsheets, end users often opt to re-develop their own implementations of existing tools, rather than use an organization’s previously built tool which may serve the same purpose. The lack of quick and easy clarity of someone else’s spreadsheet logic, especially when sheets of a higher complexity are developed, creates redundancy within an organization’s spreadsheet software repository. An alternative issue may be those issues of frequent use. The lack of a strong version control system, and/or documentation to accompany spreadsheet-based tools within the Excel framework provides a lack of clarity in where and how spreadsheets have been edited. This creates not only a lack of clarity for calculations, but a lack of clarity as to the model’s input assumptions as well. [Shubbak and Thorne, 2015]

The solutions prescribed above are not without their merits, yet each method attempts to mitigate errors derived largely from a tension between the notion that “spreadsheets are software and should be developed according as such” and the notion that “spreadsheet-authors are not programmers and would not concern themselves with such traditional development practices” [Ayalew et al., 2000]. Implicit in almost every paper is the notion that spreadsheets are used as alternatives to programming languages due to their lack of (perceived) complexity, and ease of use.

3. OPEN SOURCE AS AN ALTERNATIVE TO PRESCRIBED SOLUTIONS

Recent innovations the open source programming language, Python, have made significant inroads in bridging the gap between the software development flexibility, and interactive user-friendliness. Previously, many programming languages (VBA included) have lacked a strong, user-friendly interface for interactive computing. The development of an interface called the Jupyter Notebook enables the user to run calculations in a simple, documentable fashion. While the topic of this paper will later focus on the use of the Notebook platform to develop an oracle, further discussion of open source technology is warranted to provide a complete discussion to the previously discussed solutions in section 2.

Shubbak and Thorne describe centralization as one possible technical solution to mitigating spreadsheet error risk, however they state that this may incur risks of its own owing in part to new environments being unsuitable to artefacts in most organizations. Additionally, they describe how when a spreadsheet is used and modified by multiple users, the risk of error increases due to the possible changes performed on the sheet [Shubbak and Thorne 2015]. Implicitly, this is a version control issue: the Excel environment lacks a strong system for tracking and managing changes. In the programming community (open or closed source), centralization and version control are the rule rather than the exception. Platforms such as Github and Bitbucket allow an enterprise to publicly or privately manage their entire codebase from a central location, with changes tracked by user, often with descriptions (depending on internal policy), at every update. These de-facto practices and environments represent a significant departure from the Excel and VBA-related environments.

The previously cited papers have concluded that SSATs work to a reasonable, yet non-comprehensive degree of accuracy. As Kulesz and Ostberg note, the applications present differences in results for even simple pattern matchings due specifically to a program’s inexplicit assumptions and exceptions on how to treat various errors [Kulesz and Ostberg, 2013]. This is a problem that can be easily remedied by the open-sourcing of an audit/inspection engine to provide a base standard of unification from which third party, topic-specific solutions can be built.
from. Should such an open source solution be implemented, the userbase would additionally unlock access to the well-documented machine learning analytics packages which lay central to the Python language’s data analysis power. In doing so, programs may better develop predictions and classifications within spreadsheets, given a proper corpus of data, whether privately by an audit company, or publicly through known spreadsheet repositories.

4. METHODOLOGY

4.1 Overview of Approach & Goals

With an end goal of developing a library of Python functions with which to test future spreadsheet models against – referenced hereafter as the oracle, this paper will explore a possible path of recreating spreadsheet calculations in the Python language, and then define an approach to auditing such calculations within the spreadsheet itself. While the approach below begins by building an auditing toolkit by taking a pre-existing Excel-based oracle (one which we assume to be correct), and replicates it in Python, this paper intends to showcase the clarity of the calculations being performed in the Python language itself. After replicating and validating the model described below, this paper will intentionally introduce an error into the spreadsheet calculations, and utilize one possible approach of identifying where this error occurs.

Both files may be downloaded from the author’s website:

4.2 Example to be Used

This example will showcase time-flexible calculations of EBITDA for a solar power purchase agreement, given the following inputs:

- Monthly Avg. Solar Irradiance (kWh/m²)
- Plant Capacity/Size (kWp)
- Derate Factor (%)
- Annual Plant Degradation Rate (%)
- PPA Sales Price ($/kWh)
- Operations & Maintenance Costs ($/kWh)
- Inflation Rate for PPA Escalation and Operational Expenditures (%) 

4.3 Setting Up the Spreadsheet Model

The spreadsheet setup of this model is show below, and the following walk through will begin with the exhibit name first, followed by a description and the corresponding image.

Spreadsheet Inputs are distinguished between Model Inputs, which will apply to both the spreadsheet and the Python versioning. Spreadsheet inputs will only be required locally for the spreadsheet model. Certain inputs in Excel have been named for ease of access in Python.
4.4 Modeling the Net Power Generation

The solar irradiance table (G26:O37 below) shows the total irradiance (the sun’s resource available to be transformed into energy by the solar plant) per period by making use of an INDEX lookup function, and is multiplied by the operating mask in row 11 of the timing section (above) for each corresponding period. While laying a month-by-calculation out in a table might be unnecessary for a monthly model, these tables quickly become useful when periods consisting of multiple months (quarters, semi-annual, etc.) are to be applied. These figures are summed in row 38, and multiplied by the plant size, and by the derate factor (the loss of power conversion from DC to AC) in order to compute Nominal Generation. Nominal Generation is then divided by a Degradation Index to compute the Net Power Generation of a project.
4.5: Computation of Real and Nominal EBITDA

Revenue is calculated by multiplying the net power generation by a hypothetical sales price per kWh, and Operational Expenses are computed in the same manner. EBITDA = Gross Revenue – Operational Expenditures. To calculate the Nominal values, these figures are multiplied by an inflation index.

Exhibit 4.3: Computation of Real and Nominal EBITDA

5. AUDIT ORACLE TOOLKIT CREATION & VALIDATION

5.1 Python Package Requirements

This presentation makes use of the Python language (version 3.0 +), and three free, readily accessible packages: xlwings, NumPy, and the Jupyter Notebook interface. For users new to Python, the easiest way to access the bundle is by going to http://continuum.io/downloads, and download the relevant open-source Anaconda distribution package. Anaconda includes all the prerequisites above, and many more.

Python, and its various packages, may be best understood by first examining a familiar exhibit: the Excel Home ribbon.

Exhibit 5.1: Excel Home Ribbon

Imagine Excel if all it had were basic arithmetic capacities, and perhaps simple formatting from the home tab. The capabilities provided in the Formulas, Data and Charting tabs are all still available, only they would require separate installation. This is the case with Python. By downloading only the Python language, we have installed a platform for additional functionality. Each additional package installed adds specific functionality ranging from numerical computing (NumPy package), integration with Excel (xlwings package), and an accessible user interface (IPython/Jupyter Notebooks). Additional standard packages of note for spreadsheet users, which
will not be discussed in this paper, are the Pandas package for data analysis, and the Matplotlib package for charting and graphing.

5.2 Creating the Oracle: Defining an Approach

In the spreadsheet example above, a simple spreadsheet model was created to compute real and nominal EBITDA for the operations of a 500kWp solar plant. This was done by organizing the model into small sections of 2-3 calculations, each building off the next. This approach will be replicated as we reconstruct the model in Python’s Jupyter Notebook interface by creating a series of user defined functions with the same outputs.

5.3 Creating the & Testing the Kit Step-by-Step

*Step 1: Open the Jupyter Notebook App and Import Required Packages*

As mentioned earlier, the two key packages (also referred to as libraries) to be utilized in this walkthrough are xlwings and NumPy. xlwings enables a to link to any running spreadsheet instances, and access any of the workbook’s artefacts. NumPy is Python’s main scientific/numerical computing package, driven largely by array-wise calculations similar to those described in Swan 2016. These packages provide the basis for our calculations and modelling.

Libraries make up the open source ecosystem, and this is one of the key areas where VBA and more open source technologies begin to diverge. As the packages are imported, a “namespace” is created which enables the user to reference the library’s capabilities. In the first cell below, the libraries are imported “as” abbreviations. This is standard practice and creates abbreviated namespaces within the program, allowing the user to access each object or action within the library by calling the abbreviation.

*Exhibit 5.2: Importing packages “as” abbreviated namespaces.*

```
import xlwings as xw
import numpy as np
```

*Step 2: Create User Defined Functions*

This paper will approach modelling in Python by defining a series of functions, their respective parameters (inputs), and their returns (outputs). Math and logic come first, inputs come later. As in the Excel example, calculations are segmented into small pieces. While segmentation to this extent isn't necessary when building complete end-use models in Python, it is critical to this approach of auditing. Each of these functions can be easily reused, or combined into one, once the user is comfortable with the logic.
In addition to the function’s logic, the function’s use can be described with a callable “docstring”. If another user wishes to use the function, but is unsure how it works, the user only needs to run a command with the function name and a question mark, which will prompt a help screen with the function’s documentation. This feature applies to any documented objects, modules, or functions in Python. The ease of access of this documentation represents a significant departure from methods required by Excel or VBA.

Exhibits 5.3-5 below show the creation of a function to calculate nominal power generation with a docstring included, a call to the newly created function’s documentation, and the rest of the functions required to mirror the Excel model with comments in place of full documentation for concision.

*Exhibit 5.3: Creating a function with built-in documentation.*

```
In [2]: # nominal generation
def nom_genertion(plant_size, derate, irradiance, start_month, model_years):
    """
    This function returns the nominal AC power generation of a solar plant.
    
    Parameters
    ----------
    plant_size : float
        the size of the solar plant in terms of kW
    derate : float
        the conversion/loss factor between DC and AC energy
    irradiance : array
        the monthly solar irradiance in kWh/m2, in order from January to December
    start_month : integer
        the numerical month the project is expected to begin operations
    model_years : integer
        the modeled lifetime of the project in years
    
    return type: array
    """
    start = int(start_month - 1) # adjust for zero-indexing and integer dtype
    model_years = int(model_years)
    irradiance_ly = np.roll( irradiance, -start)
    model_irradiance = np.tile( irradiance_ly, model_years)
    return plant_size*derate*model_irradiance
```
Step 3: Define Model Inputs by Calling from Excel Workbook

- Using xlwings, ranges can be called by named ranges, cell values, or by selecting the first value of an array, and using the expand option.
- The Excel workbook needs to be open to call values from it.
- Similar to VBA, Python is "object-oriented"; there is a defined hierarchy in objects such as `xw.Book('eusprig_solar.xlsm').sheets['Model']` where the highest level, `xw` is the
namespace of the xlwings package previous imported, and Book is an object defined within the namespace.

Exhibit 5.6: Linking to Excel Workbook and Importing Assumption Values

```
In [3]:
sht = xx.Book('eusprig_solar.xlsm').sheets['Model']  # define the sheet
irradiance = sht.range('D26').options(np.array, expand='down').value  # call array
plant_size = sht.range('C28').value
rate = sht.range('C30').value
ppa_price = sht.range('C32').value
om_cost = sht.range('C34').value
deg_rate = sht.range('C36').value
inf_rate = sht.range('D45').value
start_month = sht.range('D7').value
model_years = sht.range('D8').value
ops_months = np.arange(1, model_years*12 + 1)  # add 1 due to zero-indexing
```

**Step 4: Perform Excel Model Steps in Python**

- Running the model is a matter of calling functions we created earlier.
- The Nominal EBITDA output has been printed to examine at the raw result.
- N.B. Input parameters from Step 3 (above) have been named to match the function parameters. Just as in Excel or VBA, consistency in naming conventions is of high importance.

Exhibit 5.7: Defining Output Variables by Running Functions Created in Step 2

```
In [5]:

def nom_gen = nom_generation(plant_size, rate, irradiance, start_month, model_years)
    def_index = discount_index(ops_months, deg_rate)
    nom_gen = net_generation(nom_gen, def_index)

# ebitda & inflation adjustments
    ebitda_r = real_ebitda(nom_gen, ppa_price, om_cost)
    inf_index = discount_index(ops_months, inf_rate)

# could be computed easier...
    ebitda_n = nominal_ebitda(nom_gen, ppa_price, om_cost, inf_index)

print('Nominal EBITDA: [ 7007.75117558 5829.56457816 5978.7687986 4984.34222296 4184.14120992
3699.1778833 4191.0174196 4967.5207397 5654.62931968 6469.90202742
6620.52255148 6947.10891089 7077.13485059 5887.28383933 6037.9030738
5035.6914596 4225.5683497 3735.80341198 4232.51264518 5016.70411337
5710.61574859 6533.9604633 6886.07227971 7015.89216743 7147.20549267
5945.57297041 6097.68422528 5083.53068206 4267.40566089 3772.79156457
4274.41870734 5066.37445112 5767.15649858 6598.65314119 6752.27010151
7085.55644632 7217.96990349 6004.44002952 6159.95736442 5133.80269091
4309.65720128 3810.1459365 4316.73968464 5116.5365744 5824.25705797
6663.9863406 6819.12518362 7155.50849034 7289.43495204 6063.8899308
6219.02820114 5184.69299246 4352.32707456 3847.87015369 4359.47968152
5167.19535236 5881.92296944 6729.96640388 6886.64127455 7226.35510906]
```

**Step 5: Compare and Validate Results between Python Calculations and Excel**

**General Approach:**
• Best practice: start from the last calculation. In this case, nominal EBITDA.
• Use np.allclose(), and np.isclose() comparison functions evaluate results within a set tolerance. 1x10^-5 set as default.
• Use np.where() to determine cell locations - similar to Excel's MATCH function.

Other Practical Considerations:

• The range called in Excel contains extra columns as contingencies for a delayed operating start: calculations are housed within the spreadsheet framework, whereas in Python, calculations only as long as required. To adjust for this, we need to resize the Excel array to isolate only the columns used in the calculation.
• These columns can be isolated by "slicing" larger array to appropriate size. This is done by adding brackets after an array name.
• To take the appropriate "slice", we have adjust for Python's zero-based indexing.

Exhibit 4.8: Confirming Model Equivalency and Accuracy

6. AUDITING CALCULATIONS

6.1 Finding the Error: Approach Overview

Now that the oracle has been created and validated, it can be used to test against erratic spreadsheets with the same goal. In this section, a common financial modelling error has been introduced to an undisclosed section of the spreadsheet model. This section will walk step-by-step through the oracle’s use, with help from the previously mentioned NumPy functions, allclose(), isclose() and where().

6.2 Using the Python Oracle to Find the Error: Step-by-Step

Step 0: Replaying Our Last Calculation with New Information

As seen below, replaying the previous validation command returns a False output. The calculations do not match up.

Exhibit 6.1: Retesting Model Equivalency
**Step 1: Step Backwards to Find Correct Sections**

A series of print statements enables the user to “step backwards” through the major calculation points of the model to in an attempt to isolate an area where the error has been introduced. Calculations such as inflation or degradation would be skipped in this initial section as they are additive to more major breakpoints, and would be explored only if our initial search warranted it.

**Exhibit 6.2: Testing Model Modules with a Broad Scope**

```python
import pandas as pd

# Set up data
input_data = pd.read_excel('input.xlsx')
output_data = pd.read_excel('output.xlsx')

# Perform calculations
result = input_data.apply(lambda x: x * 2)
output_data = output_data.apply(lambda x: x + 1)

# Check results
if result.equals(output_data):
    print('Calculations match')
else:
    print('Calculations do not match')
```

**Step 2: Drilling Down on Errors**

Above, the Nominal Generation calculations return True, while Net Generation remains False. This leaves two possible areas to look for drill down for errors: the Degradation Index calculation, and the Net Generation calculation.

From here, we can run two tests. As the degradation index has not yet been compared, we will call the np.allclose() function once more. As Jupyter’s interface allows for multiple commands at a time, Net Generation can be examined further in the same block further by testing for similarity between each element in the array (as seen below).

**Exhibit 6.3: Testing with a Higher Level of Detail**

```python
# Calculate degradation index
x1_deg_index = (input_data['Degradation Index'] * output_data['Index']).sum()

# Check degradation index
if x1_deg_index == output_data['Degradation Index'].sum():
    print('Degradation Index matches')
else:
    print('Degradation Index does not match')

# Calculate net generation
x1_net_gen = (input_data['Net Generation'] * output_data['Generation']).sum()

# Check net generation
if x1_net_gen == output_data['Net Generation'].sum():
    print('Net Generation matches')
else:
    print('Net Generation does not match')
```
As results above are examined, the outputs show that the Degradation Index is not the source of error. The Net Generation np.isclose() output shows that while the first “cell” in the array is correct, the rest are not. By drilling down on initial tests, the source of the error has been identified.

This error is immediately guessable: the first cell in an array was edited, but not unlikely pasted across.

**Step 3: Identifying Problematic Excel Cells**

Above, results showed that the first cell of the Net Generation array is correct, and likely not pasted across. This is easy to see visually because it occurs at the beginning of the array. But what if it was somewhere in the middle of the array? By calling the Excel range object of the array, without appending it with a “.value” method, we can use the np.where function and some indexing (as seen below in Exhibit 6.4) to return the correct and erratic cells of the array.

**Exhibit 6.4: Identifying the Error Cell Locations**

```python
In [36]: x1_n0m_gen_rng = sht.range('G41').expand('right')

#find index locations
net_gen_isclose = np.isclose(net_gen, x1_n0m_gen)
error_indices = np.where(net_gen_isclose == False)
correct_indices = np.where(net_gen_isclose == True)
print('Correct Indices: ', correct_indices)
print('Error Indices: ', error_indices)

#find the appropriate indices beginning and end
correct_lx = correct_indices
error_lx_max = max((np.amax(error_indices)), len(ops_months))
error_lx_min = np.amin(error_indices)

correct_range = x1_n0m_gen_rng[:, correct_lx:correct_lx_max].address
error_range = x1_n0m_gen_rng[:, error_lx_min:error_lx_max].address

print('Correct Cell/Range: ', correct_range)
print('Error Cell/Range: ', error_range)
```

Correct Indices: (array([0], dtype=int64),)

Error Indices: (array([1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16, 17, 18, 19, 20, 21, 22, 23, 24, 25, 26, 27, 28, 29, 30, 31, 32, 33, 34, 35, 36, 37, 38, 39, 40, 41, 42, 43, 44, 45, 46, 47, 48, 49, 50, 51, 52, 53, 54, 55, 56, 57, 58, 59], dtype=int64),)

**6. CONCLUSIONS**

This paper has sought to provide the foundations for developing an alternative to building test oracles in MS Excel by laying the groundwork in the Python language for a predominantly spreadsheet-oriented audience. Embedded in this paper is the notion that while Excel might be the current de facto tool for financial modelling, less risky alternatives exist which – when used appropriately – can replicate and build on many capabilities traditionally found in Excel.
The numerous steps shown in this paper could be safely, and concisely shortened. While the current lack of adoption among the financial community is a hindrance, the Python language presents itself as a strong alternative to the risks associated with spreadsheet model risk due to recent innovations in usability such as the Jupyter Notebook interface, and the language’s highly legible syntax. Calculations are clear, wordiness is low, and the amount of programming knowledge required to reconstruct Excel-oriented analysis tools is accessible to the average spreadsheet professional.

REFERENCES


Structuring Spreadsheets with the “Lish”
Data Model

Alan Hall, Michel Wermelinger, Tony Hirst and Santi Phithakkitnukoon.
The Open University, UK and (4th author) Chiang Mai University, Thailand.
alan.hall@open.ac.uk

ABSTRACT

A spreadsheet is remarkably flexible in representing various forms of structured data, but the individual cells have no knowledge of the larger structures of which they may form a part. This can hamper comprehension and increase formula replication, increasing the risk of error on both scores. We explore a novel data model (called the “lish”) that could form an alternative to the traditional grid in a spreadsheet-like environment. Its aim is to capture some of these higher structures while preserving the simplicity that makes a spreadsheet so attractive. It is based on cells organised into nested lists, in each of which the user may optionally employ a template to prototype repeating structures. These template elements can be likened to the marginal “cells” in the borders of a traditional worksheet, but are proper members of the sheet and may themselves contain internal structure. A small demonstration application shows the “lish” in operation.

1. INTRODUCTION

Building a spreadsheet frequently involves a high degree of replication, both at the level of cells containing the same or similar formulae and at the level of higher structures such as families of similar tables. In software engineering terms this is a contravention of the “don't repeat yourself”, or DRY, principle [Thomas & Hunt, 1999]. It increases the risk of errors due to possible inconsistency among the repeated elements, and can make maintenance particularly problematic.

There is a long history of developments aimed at capturing some of the higher level structures to be found in spreadsheet models, reducing the need for replication. Notable early efforts were Lotus Improv (in 1991), which introduced column formulae and named dimensions; and the pioneering work of [Burnett et al., 2001] whose “dynamic grids” allowed tables (and regions of them) to be manipulated as discrete objects. Excelsior [Ireson-Paine, 2005] banished the replication entirely, at the expense of requiring the user to write code in a spreadsheet description language. More mainstream examples include the availability of array formulae, pivot tables, and ranges designated as “Tables” in Excel.

One potential barrier to uptake of more structured approaches like these is that they require the user to master new abstractions over and above the engagingly simple one that is the standard spreadsheet grid. Our approach is therefore based on a modification of the grid itself – a new abstraction of a kind, to be sure, but a rather simple one – as opposed to any new apparatus to be supplied alongside it. We borrow from [Erwig et al., 2006] the notion of a template that can generate a structure, but do so on-sheet and with no prior assumptions about what forms of table are legal.

Recently this area has seen a flurry of activity, partly arising from the application of
spreadsheet-like approaches to relational database queries [McCutchen et al., 2016; Bakke & Karger, 2016; Chang & Myers, 2016]. We place our work more squarely in the domain of general purpose spreadsheets, along with [Hodnigg & Pinzger, 2015] who developed procedures for identifying areas of the worksheet by role and similarity of formulae, in order to visualise sets of cells with some commonality as a single cognitive unit. Our model encourages a structure in which such cognitive units are expressly defined. We also draw on the work of [Miller & Hermans, 2016] and in particular their notion of a “semantic axis” which has parallels with the way we use our template cells.

In tackling these issues at the level of the underlying data model – our grid replacement – we are proposing an organising principle for spreadsheet-like data, rather than a blueprint for a tool as such. With the right choice of model, we can make certain common spreadsheet structures (e.g. multi-level tables, and families of similar tables) emerge naturally, as opposed to requiring new abstractions. So ours is a complementary approach to the tools mentioned above, some of which might sit more comfortably upon this data model than upon the traditional spreadsheet grid.

We have implemented our data model, called the “lish”, and its associated operations in a small demonstration application. It is work in progress but currently supports basic editing and interactive transformation of spreadsheet data into lish form, though not (as yet) calculation. It includes a layout engine that forms the basis of a fledgling GUI and which was used to produce the screenshots in Figures 2.2 to 2.4. It reads and writes data as JSON.

In the next section, we explain what our new data model consists of by means of some example spreadsheets, and in section 3 we summarise the underpinning theory. Section 4 wraps up with our planning for the next stages of this research.

2. AN ALTERNATIVE TO THE SPREADSHEET GRID

2.1 Row and Column Headers are Part of the Worksheet

Figure 2.1 shows a small example spreadsheet (produced in LibreOffice) which takes some population and geographical area data and calculates population density by region. The cells in the two population density columns each contain a formula which references the area and the relevant annual population column. In the figure, one such formula is visible in cell E8.

We begin our reformulation of the spreadsheet grid with a very simple, almost trivial change. It concerns the row and column headers: those marginal “cells” (see Figure 2.1) containing the row numbers and column letters respectively. In a normal spreadsheet they form a labelling area and are not really proper “cells”, but our first change is to make them so. We allow the user to select and edit them, just like any other cell. The expected (though not mandated) usage is that the user will position column headings in these marginal areas, rather than in the internal cells of the worksheet.

In a normal spreadsheet, the user may set properties (such as numeric format) across an entire row or column by clicking in the appropriate header. Our equivalent is to allow the user to set properties on the equivalent marginal cell, whereupon they are inherited by the corresponding row, or column.
2.2 Internal and Nested Margins

We pursue this idea a little further by allowing the user to define further ranges within the worksheets — somewhat like conventional named ranges, but in our model the name is optional. Each of these ranges now acquires its own margins, which behave like those of the global worksheet described above but apply only to their local range. Figure 2.2 shows our alternative representation of the data above, in which a number of such ranges
have been defined: there is one for the small independent table at the top (the metadata giving the scope with regard to age and sex), and a larger one for the main table. The example also shows a further extension to this scheme, namely that ranges may be nested inside one another. The main table is in this way partitioned into three internal sub-tables: the left hand one contains the Region and Area columns, the middle one contains Population and the right hand one, Population Density. The shaded cells are all marginal, in the sense described above, and the nesting is visually apparent from the double margins on the main table – one margin for the top level table, and one for each nested range within it.

2.3 Cursor Behaviour

It will be noticed that in Figure 2.2, three cells are selected by a red wireframe cursor: the population figures for the two years for “East” region, and the corresponding internal marginal cell. However this is not an arbitrary selection of the kind one would obtain by dragging across a range of cells in a conventional spreadsheet. Rather, the selection cursor in our scheme is allowed to select any of the ranges that we set up above in their entirety, or any row, column or (of course) single cell within such a range. Pressing the up-arrow in our demo application moves the cursor to the corresponding three cells (including one marginal) for “West Midlands” – it does not collapse the selection back to a single cell. The user can drill in or out of the ranges they have defined and hence navigate at varying granularity.

Our rationale for this behaviour is that when building a spreadsheet model, users often wish to perform some operation uniformly upon a selection of cells – but not just any selection. The kinds of selection that are needed very often correspond to one of the ranges which a judicious application of this scheme would generate, or a row or column within one of those ranges. Allowing the cursor to behave in this way facilitates forming such cell selections and reduces the risk of “out-by-one” errors that an imprecise mouse movement might cause.

2.4 Marginal Formulae

Just as the inputs to a formula often consist of the kinds of cursor selections we have just described, so too do the outputs. That is, in a spreadsheet we often have a range of cells containing a formula that is identical except for some regularly updating pattern in the cells referenced. This formula is only required to be located in one place; we propose that that place should be the marginal cell which governs the range in which its results are to appear. This is similar in concept to the array formulae of traditional spreadsheets but more flexible, as it is perfectly legal for insertions or deletions of cells to take place within the output range, and the idea generalises to more complex structures as we shall shortly explain.

As an example, in Figure 2.2 we could locate column formulae for the two columns calculating population density (from population count and hectares) in the top marginal cells of their respective columns, assuming that the marginal labels are separable from the formulae. Better yet, we might observe that this formula is actually the same in both columns and enter it only once, in the cell marked “X” in the figure, which forms the pivot of the population density sub-table. Note that the marginal cells themselves are to be excluded from any arithmetic (e.g. column sums) so may be included in a selection with impunity.

In a strong version of this scheme, we might even go so far as to constrain the user to
place formulae only in marginal cells. For any formula that is a one-off, yielding a single scalar value, the user would be required to create a mini “table” containing one marginal cell to contain the formula (and optionally, a label) and one ordinary cell to contain the answer. A weaker version is to allow individual cells to override their parent margin, with a warning indicator when this is taking place. Similarly, a column formula will override a range formula – the “innermost” one wins – also with a warning. In the strong version, we would also forbid arbitrary cell selections: the only multiple selections would be those corresponding to some margin. This would greatly reduce the risk of inconsistent formulas, though clearly with a trade-off against flexibility.

2.5 Margins as Templates: Generalising to Higher Dimensions

In order to follow our scheme to its logical conclusion, we will now make a shift in perspective: we will continue to deal with internal margins and with ranges of the worksheet, but we will view our margins as templates for the body of the range which they govern. The header row of a table, after all, is providing a description for the rows that follow: it says how many cells are to be in each row, i.e. how many columns there are, and by providing labels it tells the user what the meaning of those cells is. There are fewer use cases for the left hand margin performing a similar role, but in principle it could do so: for example, a cell in this margin could specify a highlight property to be applied to the entire row, and that too could be viewed as a form of template where the first cell acts as a basis for the others. A null template – i.e. an empty cell in the margin – may be interpreted as imposing no prior expectation on what the subsequent cells can contain.

![Figure 2.3. Internal structure in a margin defining a three dimensional array. All the grey cells are marginal, with the level of shading denoting depth within the structure.](image)

We have stated that marginal cells are to be editable just like any other cell. So, if we can define ranges anywhere within the sheet, what would be the implication of defining them within the margins themselves? From the template perspective, the answer is that the ranges governed by those margins must conform to whatever structure the margins define.

One application of this pattern is the formation of three dimensional arrays. Figure 2.3
shows some rainfall data broken down by quarter, year and city. The top band of shaded cells is itself a two-dimensional table. Since this band is marginal it contains no actual data but rather acts as a template, forming the base plane for a stack of tables of the same structure. Hence, we have our three dimensional array. In the figure, the user has positioned the cursor at cell level in the “Q2” heading of the template. Our application has automatically highlighted a secondary selection comprising all those cells which are governed by this marginal cell. Had the user selected the “Q2” in the internal margin belonging to Cardiff, only the two rainfall values relating to Cardiff in that column would have appeared in the secondary selection. Recall that marginal cells are omitted from calculations, so this selection could be used as the input to a formula producing summary statistics on rainfall.

Figure 2.4 shows a similar three dimensional example, but this time we do not have a “pure” three dimensional array but rather a family of objects, each containing an assortment of fields including a two dimensional table. The data are from a spreadsheet that a (fictitious) company might use as part of workforce planning. For a number of sites, the sheet shows a site id, site address, and a table of headcount broken down by grade and year.

The margin-as-template principle lends itself easily to this pattern. The object format is defined in the top shaded band of cells, and this governs a range of objects each conforming to that format. In the figure, the user has placed the cursor at the “2016” cell of the “staff” table. As in the previous example, the system has automatically generated a
secondary selection comprising the cells governed by this margin. This time, however, the secondary cells are not all adjacent – we have a cross-cutting selection containing the staffing figures for all sites and all grades, for the year 2016 only.

The data presented in this example could alternatively be held in a normalised database. The structure we have presented here captures some of the discipline associated with that option while presenting the data in a more human readable form. The marginal cells allow some of the cross-cutting selections that would otherwise require a separate query or pivot table to be generated on the data in-place, in a visually intuitive way.

Higher dimensional structures may be obtained if necessary by a generalisation of the same procedure: the outermost margin is set up as a table of tables of tables, to whatever depth is required.

2.6 Dynamic Cell Allocation

Careful examination of Figure 2.4 will reveal another feature of the way we interpret template elements: a single cell within a template is allowed to spawn an embedded range of cells at the corresponding position within the range that it governs. The site address is an example of this. In the template, only a single cell represents the site address. But for each individual instance of a site, a one-dimensional range of cells has been embedded at that location in order to accommodate an address with a variable number of lines. Our layout engine is conversant with this behaviour and accommodates the variable object size appropriately, while maintaining alignment over elements that form part of some larger table.

Our wider intention in defining this behaviour is to allow formulae to return results whose number of cells is not known at design-time. Hence for future work, we propose this as an approach to addressing the dynamic allocation problem for spreadsheets.

2.7 Dispensing with the Global Grid

We claimed in the introduction that our data model would be an alternative to the normal spreadsheet grid, as opposed to an additional construct to be overlaid upon it. We have framed the discussion above in terms of “ranges” of cells selected over the normal grid, as that is the way one would go about refactoring an existing spreadsheet into our form. But once we have it in this form, we can indeed dispense with the concept of an underlying grid altogether. All that we require is the nested system of cell ranges with their margins. Similarly, if starting from a blank document, we could create the various templates and enter the data without ever going via a global underlying grid at all.

We note here that eliminating the global grid has cleared up a minor visual annoyance with the normal spreadsheet: when two tables are vertically juxtaposed on the same worksheet, they are forced to share common column widths, even though the content in some column of the upper table might be suited by a very different width to the corresponding column in the lower one. In Figure 2.2, our layout engine has recognised that the small table at the top is a separate range from the main table, and has broken this unwanted coupling.
3. THE “LISH” DATA MODEL

This section discusses the theory of the underlying data model needed to support the behaviour described above. Formal specification is deferred to a future more technical document.

At first glance, it might seem as if what we have are simply cells within cells – a normal spreadsheet grid in which some cells – we might call them “super-cells” – are allowed to contain other cells. To a first approximation this (rather obvious) extension to the standard spreadsheet is indeed what we propose, but it's not quite the whole story. The problem is that we want some of the “super-cells” to contain grids that are the same size and in alignment with their neighbours, or even with other “super-cells” elsewhere on the sheet entirely, whereas others are under no such constraint. We would like a simple way to capture that without introducing a spaghetti-network of explicit links.

The model that we propose instead is called the “lish”. (The name is a portmanteau of “list” and “hash table” since in its fully specified form it shares some characteristics of both; it is the list-like properties that we focus on in this paper.) A lish is a list of elements, each of which can be either a single atomic value (numeric, string or boolean) or a further lish. The atomic values correspond to individual cells and the sublists to the ranges as described in the previous section. For a list to be a valid lish, it must meet certain criteria to be described shortly.

To develop the theory we will set aside the graphical world of the spreadsheet and fall back on a simpler textual notation. We could represent a small subset of the earlier population data using the following list of lists:

```
[[null, “Region”, “Area”],
 [null, “North East”, 857000],
 [null, “North West”, 1411000],
 [null, “Yorks & Humber”, 1541000]]
```

Following common programming languages, we have used here a notation where lists are enclosed in square brackets with their constituent elements separated by commas. A null denotes an empty cell; the four nulls above correspond to the cells in the left hand margin of the table in Figure 2.2. “Region” and “Area” are column names and hence occupy cells in the top margin. In this example, there are no row names, so the left hand margin contains only nulls.

We want the list of lists above to represent a table: the third element of each sublist, for instance, is related to the third element of every other sublist. Taken together, these elements form the Area column. But there could be other lists of lists where there is no tabular interpretation, and the third element in one list is not related in any particular way to the third element in any other. We need a way to distinguish these cases. This could be achieved by the use of markup tags (as in HTML) or validation against an external artefact (as in JSON Schema) but we prefer here to distinguish them by means of the list structure itself. Our rationale is that in a spreadsheet, the user creates a structure simply from the shape of the data as entered. If we can make the supporting structure self-describing, we avoid the ceremony of having to tell the system how it is supposed to be interpreted.

The previous section introduced the idea of marginal cells being regarded as “templates” for other cells in their range, and this is the approach we take to constructing the system
of nested lists that is to represent them. Every list that is a *lish* must contain at least one element – there is no empty lish. This first element in each lish is used to represent a marginal cell. It does not contain normal data, but instead performs the template role for subsequent elements in the same lish.

The rule, stated informally, is that the subsequent elements must contain “at least as much structure” as their template, but are allowed to contain more. Everywhere the template contains an atom, the subsequent elements in that lish may contain either an atom or another lish; everywhere the template contains a lish, the subsequent elements must also contain a lish, and it must be of the same length.

For example, the following lists are valid lishes:

- \([\text{null}, 1, 2, [3, 4], 5]\)
- template is atomic, subsequent elements are mix of atomic and list
- \([[\text{null}, \text{null}, \text{null}], [\text{null}, 1, 2], [\text{null}, 3, 4]]\)
- template is a list of three, so are subsequent elements (a table)

But the following lists are not valid lishes:

- \([[\text{null}, \text{null}], 1, 2, 3]\)
- template is a list, but subsequent elements are atoms
- \([[\text{null}, \text{null}], [1, 2, 3]]\)
- template is a list of two, but next element is a list of three

The above rule is applied recursively, within both the template and the subsequent elements. So a template element of \([[\text{null}, \text{null}, \text{null}]]\) at the head of some lish would be illegal: the template now has a template of its own, a list of two, which the final null does not respect. \([\text{null}, [\text{null}, \text{null}]]\) would be permissible since an atomic template may be succeeded by a list. Some more examples illustrate further the recursive application of the template rule:

- \([[\text{null}, [\text{null}, \text{null}]], [1, [2, 3]], [4, [5, 6]]]\)
  - *legal*, all three elements of the outer lish have the same form
  - and all are legal lishes internally
- \([[\text{null}, [\text{null}, \text{null}]], [1, [2, 3]], [4, [5, 6, 7, 8]]]\)
  - same as above except for the final \([6, 7, 8]\)
  - and *legal*: the \([6, 7, 8]\) has an atomic template, so a lish is allowed here
- \([[1, 2], 3, 4]]\)
  - a null template followed by a lish is valid
  - but *illegal*, since the atom, 3, following \([1, 2]\) is not

This rule enables us to define a number of commonly occurring structures in spreadsheets. For example, the inner top margin of the main table of Figure 2.2 is the lish \([\text{null}, \text{null}, \text{null}, [\text{null}, 2014, 2015], ["X", 2014, 2015]]\). This sets the template for subsequent rows of its table as three columns, followed by two column groups each containing a further three columns. A regular three-dimensional array is defined by a two-dimensional table within its template. For example, the rainfall data of Figure 2.3 contains within its top margin the lish:

\[
[\text{null}, 2015, 2016],
["Q1", \text{null}, \text{null}],
["Q2", \text{null}, \text{null}],
["Q3", \text{null}, \text{null}],
\]
Since the template position may contain any lish, we are not restricted to regular arrays. For example, the template for the workforce data of Figure 2.4 is:

```javascript
[null,
 null,
 ["site id", null],
 null,
 ["site address", " "],
 ["staff", 2015, 2016, 2017],
 ["assistants", null, null, null],
 ["supervisors", null, null, null],
 ["managers", null, null, null]]
```

The lish itself is agnostic to whether tables are represented row-wise or column-wise. By default, the layout engine simply alternates the orientation of sublists by depth of nesting, but this may be overridden on specific lishes according to user preference.

Of course, from the user's point of view all this is done visually by selecting ranges, which then acquire margins, and embedding further ranges within those margins when more complex templates are required: the user is insulated from the nested bracket notation entirely. But underneath, ranges are lists, individual cells are atoms, first elements of lists are marginal cells, and the template rule as stated ensures that structures respect their template while allowing additional embedded structure (new sublists) to be added over and above that template. The dynamic allocation behaviour noted in section 2.6 arises because it is legal to associate an atom in a template with a list in a cell that it governs. We use an algorithm that parses the lish structure in order to associate each marginal cell with the appropriate set of cells within the table body, yielding the “secondary selections” of Figures 2.3 and 2.4. The visual layout engine uses a similar algorithm (though not identical, since its output is always a two dimensional projection) to determine which cells are in equivalent rows or columns to others and so need to be kept aligned.

It should be noted that “atomic” elements within the template can actually perform at least three roles: they can contain labels; they can define formulae; and they can specify formatting information. So they are in fact more complex objects than the simple textual representation above would imply, but they are regarded as atomic with respect to their role in defining further structure.

There is a caveat when there are nested template elements: it can sometimes arise that a subsequent element is apparently governed simultaneously by two or more conflicting templates. A typical use case would be when we wish to group both rows and columns within a table. An extension to the theory, outside the scope of this paper, accommodates that situation by “composing” the rival templates: briefly, the element under contention acquires structure from all its templates combined, but without duplicating any structure that had a common origin. The mathematics of specifying the lish becomes rather more involved as a result, but the intuition that the first element of each lish governs the form of its successors holds good. The algorithms used in the prototype application accommodate such cases.
4. DISCUSSION

4.1 The Lish in Summary

We have proposed an alternative data model, the “lish”, that could be used as an underlying representation for a spreadsheet, in place of the traditional grid of cells. The model is based on nested lists of cells, with a constraint that enables certain structures that encompass parallel lists to be captured.

The model maps to a user's point of view in which marginal cells acquire the same capabilities as any other cell, and behave like templates over those ranges of cells which they govern. Moreover the user may define internal ranges, with their own internal marginal cells, and may even define these within the margins themselves giving rise to more complex structures.

We have shown how from the simple ground rules upon which the lish is defined, a number of patterns frequently arising in spreadsheets (including tables with grouped rows and columns, higher dimensional tables, and families of similar objects) can be made to emerge. The nested structure allows these objects to be navigated at multiple granularity, and the association between marginal cells and sub-ranges allows formulae to be defined in a way that minimises replication: the spreadsheet can be made DRY.

4.2 The User Perspective

The benefits of having a lish structure once it is in place are clear enough: users can work at multiple granularities, easily select and manipulate logically coherent portions of the worksheet, and avoid both the tedium and the risks associated with manually replicating tables and formulæ. But if users are to gain these advantages, they must be persuaded of the merits not only of learning to use a lish representation (though that would be true of any new system) but also of investing the effort in applying the extra structure to their data by means of defining ranges and associated templates. Will they perceive it as worth their effort to get there?

A full discussion of this question must await the outcome of the user studies (see next subsection). But we have already taken some steps to smooth the path for users as much as possible. In the spirit of the traditional spreadsheet, we do not impose a large number of new concepts upon the user: the large variety of structures that may be formed all arise from two basic ones, the “ranges” and “templates”. The former will be familiar to anyone who has already worked with named ranges, and the latter are not conceptually distant from the usual working pattern of designing a small part of the spreadsheet and then copy-pasting to replicate it. Working with template cells feels very much like working with normal cells – they are not a separate type of object.

Another way we stay in the spirit of a normal spreadsheet is that structures are formed “by demonstration” from the shape of the data. Users are not forced to specify their application up front before the system will allow them to do anything. They can start from a regular grid (which of course is itself a perfectly valid lish) and apply the more detailed structure upon noticing which parts of their data belong together and would be worth enclosing in a range, and where there are repeating patterns that would be worth recycling as templates.

In future work, we propose to make this process of refining the structure more attractive
to the user, by aligning it as closely as possible with the secondary notation that they might well wish to apply anyway. Referring back to Figure 2.1, the user has clearly decided to demarcate the main table by enclosing it in a box, to identify the column headings by shading them, and mark certain columns off into groups by means of vertical lines. These activities are at least closely parallel to the ones that need to be applied in order to create a lish structure, and we propose to exploit that parallel – perhaps even to the point that explicit formatting becomes deprecated in favour of a complete concordance between format and structure. An analogy might be the use of named styles in a word processor to make explicit the link between visual appearance and document structure, as opposed to having the user apply formatting piecemeal.

One further consideration for the user is conservation of complexity: merely expressing a complex structure in lish form does not simplify it. What it does do, however, is take certain constraints that in an ordinary spreadsheet would be implicit, or exist only in the user's head, and crystallise them. For example, if several tables are all supposed to perform the same type of calculation then this is a feature of the specification that always exists, whether the data are in lish form or not. But if they are, this part of the specification has been baked in and will be more robust to future maintenance. So although the complexity has not been removed, its associated risks have been mitigated.

4.3 Planned Future Work

A small prototype application is at an early stage of development. It is capable of carrying out some basic data editing and refactoring tasks interactively, but does not yet support actual calculations. We do not yet support separate worksheet tabs, but if implemented these will be treated as a visual presentation device only, so that the entire workbook is one lish. This would enable template tabs to control groups of further tabs without modification to the theory of section 3.

We are planning two trials with live users. The first will be to assess a range of real-world spreadsheet applications against the lish to see whether the intended benefits (especially around the capture of repeated structures and formulae) can indeed be realised. The second trial will take place once the prototype has been sufficiently developed to support the entire workflow of building a spreadsheet application from scratch. In this trial, users will complete tasks using the prototype with data where no lish structure has initially been applied. This will assess whether they are able to grasp the concepts of ranges and templates and apply them correctly so as to reformulate initially unstructured data into lish form.

The prototype has no support as yet for actually calculating anything. The calculation model will be formula based, as in the spreadsheet, but needs to operate not only on individual cells but at aggregate level. A “lish calculus” will therefore be specified. This will draw heavily on the use of vectorised calculations, as used in the R programming language [R Core Team, 2017], but will extend this concept to work with a general, nested, lish structure as opposed to only a one-dimensional vector or flat matrix.

ACKNOWLEDGEMENTS

The authors would like to thank the Government Operational Research Service (UK) for supporting this research.
REFERENCES


ABSTRACT

Based on the Spreadsheet Competency Framework for finance professionals, in the present paper we introduce the Edu-Edition of the Spreadsheet Competency Framework (E²SCF). We claim that building spreadsheet competences should start in education, as early as possible, and this process is a lot more effective if support arrives from expert teachers. The main feature of E²SCF is high mathability computer-supported real world problem solving. This approach is based on – from the very beginning of training – a two-directional knowledge transfer, data and error analysis and handling, and the programming aspect of spreadsheets. Based on these features, E²SCF is set up for basic and general users to build up firm spreadsheet knowledge and to develop transferable problem solving skills and competences.

1. INTRODUCTION

1.1. Twenty Principles and the Spreadsheet Competency Framework

Two fundamental documents dealing with spreadsheets were introduced and published in the last two EuSpRIG conferences: “Twenty principles for good spreadsheet practice” and “Spreadsheet competency framework – A structure for classifying spreadsheet ability in finance professionals” (SCF) in 2015 and 2016, respectively. However, like any pioneer work, these theses are in need of several revisions until a consensus is reached in the community. The second edition of “Twenty principles for good spreadsheet practice” was already published and welcomed in 2016. In the present paper we provide the details of our remarks, comments, and suggestions for reconsidering the SCF, and present the Edu-Edition of the Spreadsheet Competency Framework (E²SCF).

Our suggestions for the revision are based on the role, importance, and influence of teaching spreadsheets, focusing on knowledge transfer between the different subfields of computer sciences and informatics (ICT for short) and other traditional sciences – closely related to the teachers’ beliefs in the incremental nature of science –, the influence and role of expert teachers (Hatie, 2003; Chen et al., 2015; Csernoch, 2017), computational thinking (Wing, 2006), schema-construction (Skemp, 1971), and the educational aspect of end-user studies and training in general.

1.2. About the Edu-Edition of the Spreadsheet Competency Framework

The primary feature of this edition is that general end-user training and education have been inserted into the spreadsheet framework. We formulated the educational version on the basis of the original Spreadsheet Competency Framework. In this context we focus on the triad of (1) computational thinking, (2) knowledge transfer, and (3) computer problem solving (CT–KT–CPS), how these phenomena strengthen each other, and can be placed in spreadsheet environments.

The widely accepted “classical” approach is that “[t]he typical spreadsheet user just wants to use a power tool, and the craft comes later.” However, both our research in spreadsheet...
training/education and research in education-effectiveness have proved that those who are self-taught or have learned spreadsheets separately from other birotical software (‘bureautique’ or office applications) or further ICT environments, or focus on the use of the spreadsheet interfaces and/or tools, do not develop the “deep understanding of the science”. This means that their knowledge is limited in the sense that it is not open to further input, cannot be transferred to other environments, and is hardly open to further development. With this approach the “sunk-cost” fallacy also takes its toll. It means that once end-users and even teachers make some progress armed with “the belief in the fixed nature of science” they are reluctant to try and learn approaches which might require computational thinking – a method more demanding at the beginning than “click here and there” training – but more effective in the long run. To re-train those who learned the less effective methods is highly demanding, both mentally and financially.

Instead, we suggest an early start with the CT–KT–CPS triad, which can widen the participants’ perspectives and their understanding of the science, as well as building firm fundamental spreadsheet knowledge.

1.3. “Classical” methods vs. Functional Modelling

The other framework into which the CT–KT–CPS triad can be fitted is Functional Modelling (Hubwieser 2004; Schneider 2004, 2005). Based on this model, functional programming – programming in general – is highly supported, and spreadsheets are used and taught as a simplified language. Sprego – Spreadsheet Lego – is placed in this framework, with its further simplifications, which match the requirements of introductory programming.

Both the “classical” and the CT–KT–CPS approaches have their advantages and disadvantages. The advantage of the “classical” approach is that training focuses on special features, and trainees can be framed into interfaces to gain this special knowledge in a reasonably short period of time. However, as has already been mentioned, the disadvantage of this method is that this knowledge is difficult to transfer, does not support schema construction, and consequently, knowledge is not built up in long term memory (Csernoch & Biró, 2013, 2014; Biró & Csernoch, 2014). In short, end-users forget what they learn, and they have to be provided with training on the same level several times in order to acquire some reliable knowledge. In the long run this approach is time consuming and highly demanding, considering both human and technical resources.

The disadvantage of the CT–KT–CPS triad is that it takes time. However, this is normal, since it develops fundamental skill – computational thinking (Wing, 2006) – which is as important in the digital era as the classical skills: reading, writing, and arithmetic – known as the 3Rs. In the long run, developing basic skills are more important than short term interface and tool management.

The development of basic skills should be started as early as possible. However, the approach can also work with adults. Our research has proved that young adults in tertiary education at first are reluctant to accept this novel approach (the “sunk-cost” fallacy in action, again). However, we have found that their results improve significantly after they make progress in developing their computational thinking and algorithmic skills.

Furthermore, in general education we cannot only focus on those students who are oriented towards engineering, scientific research, medicine, etc. With our model we would like to reach an even wider public, even those who have no mathematical background beyond the four basic operators. This is one of the reasons that we fundamentally changed the introductory methods, and focus on text-based problems and tasks, instead of numerical examples. We have found that using non-mathematical problems in our training is a lot more motivating and they are as good as other problems for developing the algorithmic skills of the students. Furthermore, non-mathematical
problems support computational thinking, computer problem solving, and human-computer interaction better than pure mathematical and financial problems.

2. Analysing the “Spreadsheet competency framework”

The SCF is mainly built for the closed community of finance professionals, despite claims to the contrary (SCF, 2016, p. 2, p. 3; Section 2.1). This community is closed in two senses: (1) the group of finance professionals is only a small, well defined proportion of spreadsheet users with clearly distinguished and distinguishable skills, and (2) the SCF does not leave space for the further development of these end-users. Beyond that, the document does not consider spreadsheet management as part of the greater ICT community (Section 2.1), which would be crucial in the acceptance of end-user computing (Panko, 2013, 2015; Panko & Port, 2013; Kadijevich, 2013). In general, this approach leads to a situation in which the SCF and spreadsheet management are secluded, both from the digital and the traditional sciences.

2.1. The SCF: the “Preface” and “About this framework” Chapters

The preface of the SCF clearly presents two of the greatest problems related to spreadsheet knowledge and competences (SCF, 2016, p. 2): (1) “Spreadsheet skills are often learned ad hoc – almost two-thirds of Excel Community users are self-taught.”, (2) “Many users are unaware of their own true competency. Novices are generally overconfident; experts tend to sell themselves short.”

Furthermore, four levels of users are defined and described in this introductory section (SCF, 2016, p. 3): basic users (BU), general users (GU), creators (CR), and developers (DE).

The original SCF claims that “…while the levels are designed with a finance function in mind, their content is largely applicable to any person that uses spreadsheets in their job.” However, we have found that the SCF is effectively narrowed down to financial purposes, indicating a closed type of spreadsheet usage, instead of emphasizing the openness of the software and the wide range of possible goals.

2.2. The SCF: Chapter “The framework specification”

The core of the SCF is “The framework specification”. This chapter consists of a table of 1 + 4 columns (SCF, 2016, p. 5–7). In the first, non-titled, column a rather random selection of tools, skills, competences, and activities are listed and named as items. Neither the selection criteria of these items nor their presented order are known. At Level 1 ten major categories are named and each main category contains further subcategories. The other four columns are reserved to separate the levels of users. A cell is identified by the section of an item and a level. The character of the cell indicates whether that item is required or not at that level. For each item and level three options are available:

- if the cell is empty, the item is not required at that level,
- a white circle indicates the core items (in the present paper the • character),
- a black circle indicates the beneficial items (in the present paper the ○ character).

The white and black circles play a crucial role in the framework, since they indicate the requirements clearly. In the case of the core items, this is knowledge the users must have, while in the case of the beneficial items, it is knowledge they should have. However, according to the document, the phenomena which are substituted by the expression “item” are interchangeable, regardless of their original content. It is also clear that more tools are listed than skills. Furthermore, competences, which should be the focus of the work, according to its title, cannot be clearly recognized in the document.
We also suggest a change in the characters used for the core and the beneficial items. The filled circle suggests more requirements; it is more emphatic than the empty circle. In the present paper the original white circle is substituted with the ●, while the black with the ○ character.

**Selected examples of misjudged items**

In the following, we list various items and their levels, which we consider to be misjudged and/or misplaced in the SCF. Due to the size restriction of the present paper, we primarily focus on those items which are the subject of knowledge transfer and which can be moved to basic and/or general levels in the W^2SCF. This schema focuses on the development of computational thinking, which is a basic skill which all professionals should have in the digital era, and on which they can base the special requirements of their professions. (The items are presented with their main and subcategories with the → character between them, while the levels are indicated with the –, ●, ○ characters.)

| Item: Data analysis → Excel tables → Use data stored in an Excel table |
| Level: ● – ○ GU, CR, DE |

It is not clear what content and competences are connected to the expression “Use data stored in an Excel table”. Spreadsheet management is about using data stored in these tables. Consequently, several different competences on the different levels can be assigned to this item. However, to go into details, considering all the competences in connection with “use data” is far beyond the scope of the present paper.

| Item: Data analysis → Excel tables → Use Excel tables to manage data |
| Level: ● – ○ CR, DE |

Similar to the previous item, all the levels would use spreadsheet tables to manage data. For basic users, these include simple or well protected tables, and for higher levels more complicated tables with more demanding tasks to solve.

| Item: Development and problem solving → Break down and research problem |
| Level: ● DE ○ CR |

In our context “problem solving” means solving tasks, formulating and answering questions based on the available data or on the requirement(s) of the user. Beyond that, in our framework computer tools – both hardware and software – play a fundamental role; ultimately they provide the tools to solve the formulated problems. “This usage has basically two forms: in some cases we use existing functions and methods provided by a system, and we apply these tools to solve the problems. Another possibility is, if we, based on existing means of the system, develop new programs and functions for solving new problems.” (Baranyi & Gilányi, 2013), i.e. low- and high-mathability problem solving, respectively (Biró & Csernoch, 2015a, 2015b).
Handling errors is only one element of general problem solving. This special section is part of the discussion and debugging process of high-mathability problem solving. Even for the simplest problem, and consequently for the lowest level of users, problem solving skills should be a fundamental requirement. Without analysing the problem users should not start using any software. Aimless wandering around the interface usually leads to unreliable output.
Discussion and error recognition should be part of the activities at any level. Just like the previous item, this must also be present at the lowest level. Even the simplest data recording activities cannot be managed safely without tracing errors. Users have to be trained from the very beginning that they should be aware of all their activities and their consequences.

**Examples of miscategorised items**

In the following we list items which are referred to as spreadsheet competences, but which are not specifically spreadsheet but general ICT or other traditional subject-based knowledge.

**Item: Basic skills**
- Access and save files, Read and enter data, Set up and printing
- Level: BU, GU, CR, DE

All these three items are general ICT skills and should be transferred to spreadsheets; consequently, they should not be listed as spreadsheet competency items.

**Item: Design and best practice**
- File naming and version control
- Level: DE, GU, CR

“File naming and version control” is closely related to the three previously mentioned items. Similar to them, handling files is a general ICT skill, and moreover, “version control” is extremely software-specific. For a deeper understanding, instead of “version control” the more general “conversion with Save as” would be preferable as a core item on all the levels. This option includes “version control”, and in addition, knowledge transfer would serve users’ interests better. Furthermore, “conversion with Save as” would lead users to file conversions, types, extensions, and file naming in general, which is a reversal knowledge transfer (discussed in Section 3.1), although it is beyond the scope of the present paper.

One further remark in connection with “version control” is that it is extremely time consuming and rather challenging to follow all the changes which MS applies to Excel. With 500+ functions, and their – not uncommonly – ambiguous lists of arguments and descriptions, erroneous and constant-based examples (Section 2.3; Csernoch 2014, 2017), there is no chance for effective and error free document handling. Beyond this, the older versions of Help are extremely difficult to access, even on the Internet, which makes “version control” even more challenging. The managing of “version control” was one of the reasons which led us to compose a simplified, version-independent, spreadsheet-based programming language, Sprego (Csernoch, 2014; Csernoch & Biró, 2015a, 2015b, 2015c, 2016a, 2016b; Biró & Csernoch, 2015a, 2015b; Section 3.2).

**Item: Efficiency of use**
- Shortcuts
- Navigation shortcuts, Find and replace
- Level: BU, GU, CR, DE

**Item: Efficiency of use**
- Shortcuts
- Copy and paste shortcuts
- Level: BU, GU, CR, DE
“Navigation shortcuts”, “Copy and paste shortcuts”, “Find and replace” can be transferred from general ICT knowledge.

<table>
<thead>
<tr>
<th>Item: Efficiency of use → Shortcuts → Level 3: Additional shortcuts</th>
<th>Level:     –</th>
<th>CR, DE</th>
</tr>
</thead>
</table>

The “Additional shortcuts” expression is so general that it should not be listed in the framework specification table.

<table>
<thead>
<tr>
<th>Item: Formulas → Text formulas</th>
<th>Level:     –</th>
<th>GU, CR, DE</th>
</tr>
</thead>
</table>

According to our ultimate goal, to set up the Edu-Edition of the framework, we must be aware that handling texts serves both beginners and students extremely well. On the one hand, novice users are more comfortable with strings than numbers, especially students of elementary and low high school classes, and users not specialized in mathematics, finance, economics, etc., On the other hand, displaying and handling characters in spreadsheets is well supported and, along with this capability of spreadsheets, text-based functions offer great support for an understanding of the different data types.

2.3. SCF: Chapter “Explanatory notes to the framework”

The explanatory section of the SCF (SCF, 2016, p. 8–18) is intended to clarify the items of the “framework specification” table and gives reasons for the inclusion of some of the items. The list of items in the “framework specification” makes it clear that the SCF tends to focus on the tools rather than the skills and competences, and this tendency is strengthened in the “explanatory notes” chapter.

The examples selected for the present paper refer to low mathability activities, where the tools are the focus, instead of the problem solving approaches, which is in accordance with the position of problem solving in the “framework specification” and the levels assigned to it (Section 2.2). Problem solving in this framework is handled as something mysterious which is the privilege of a select few.

Basic and general users are guided towards mechanical activities, and the SCF does not require them to do any creative work. However, we argue that any user level would carry out creative work within the context of their range of ability, and the SCF should define these skills and competences instead of requiring the mere use of a piece of software or a software family.

Examples of arithmetic and logical formulas in the SCF

Our research and practice proved that handling arithmetic formulas for basic and general users, as is detailed in Section 3.2, is more demanding than working with texts. The case is similar with logical operators, a phenomenon which is unknown at this level, without experience in programming. Novices understand the expression yes/no question, transferred from language studies, better than logical test/logical expression/condition. The reformulation of help expressions would also support novices in understanding programming concepts (Csernoch, 2014, 2017).

The logical formulas – condition-based problem specific functions –, listed in the explanatory section, along with the family of COUNT() functions, are the black-sheep of the spreadsheet family. While the basic IF() function is mandatory (Csérenyha, 2014; Csérenyha & Biró, 2015a, 2015b, 2015c, 2016a, 2016b; Biró & Csérenyha, 2015a, 2015b),
the other *IF?() functions are completely unnecessary and loaded with serious restrictions (Csérbó, 2014): they are version dependent; only the AND connection is defined, no formula is allowed in the test, inequality is handled as a string instead of as an operator, using variables is not supported. Furthermore, there are varying lists of arguments, unknown programming concepts in Help descriptions, categorised in different function groups, and a limited number of functions for only a limited number of problems. This latest restriction is completely “non-programming’. One of the tools with which MS has tried to make spreadsheets more user-friendly is the introduction of novel functions. However, MS does not recognize that they would never be able to introduce as many functions as necessary to solve all the problems in the world, even though requests arrive from the uservoice forum. The uservoice forum is not relevant in this context, since we argue against the “classical” ineffective training and usage of spreadsheets, whose main feature is the hundreds of functions. It is obvious that MS developed the *IF?() functions in response to users’ demand. However, we proved that these functions never reached the wider public. It was found in our testing that end-users can only handle the COUNTIF() function to some extent: with equality and constant in the condition. They do not know the name of the other *IF?() functions, how to handle inequality, and variables in these functions, even in the COUNTIF().

Instead, we argue that if we train students and end-users to apply the algorithm of these problems they would be a lot more flexible, version independent, and ultimately more effective. The algorithm for substituting problem specific logical functions – COUNT(*) and *IF?() functions – is the following: (1) formulating yes/no(s) question, (2) making decisions on the output values of the TRUE, FALSE branches, and (3) applying the instruction to the output values of the previous step (Csérbó, 2014; Csérbó & Biró, 2015a, 2015b, 2015c, 2016a, 2016b; Biró & Csérbó, 2015a, 2015b). With this algorithm only the yes/no questions have to be formulated properly and only one function, the IF(), has to be understand thoroughly. Consequently, all the COUNT(*) and *IF?() functions become unnecessary.

There is always a serious debate regarding how deep we need to go into the functional model. For example, do we have to include the AVERAGE() function or not in our basic set? If we go deeper than the AVERAGE() function, we need a method to count the number of elements involved. However, there is no need for the COUNT() function either; it can be replaced with the SUM(IF()) composite function, based on the algorithm detailed above. Beyond that the COUNT() function with all its alterations has become rather confusing for end-users not specialized in maths.

There are two further reasons which do not support extreme abysses in the functional model. One is discussed in the present paper (Section 2.2): even calculating an average with the SUM() and the COUNT() combination is too complicated maths for young children. The other reason is that research has proved that students do not have to know the theoretical background to solve maths problems, but can apply the available computer tools effectively (Chmielewska & Gilányi, 2015; Chmielewska et al., 2016). Consequently, we have found more advantages than disadvantages for including the AVERAGE() function in basic and general training.

Examples of lookup formulas in SCF

Lookup functions are crucial but problematic in spreadsheets. There are remnants of older versions, which might cause version inconveniences. However, the major problem originates in the HLOOKUP()/VLOOKUP() functions. These two functions are highly supported in spreadsheets forums, even though they are unnecessarily complicated and carry serious restrictions. The MATCH() function would be an alternative solution but both the function’s wizard and its Help facility, along with the SCF’s explanatory notes (SCF, 2016, p. 12), are incorrect and inconsistent (Csérbó, 2014, 2017).
The common errors related to the lookup functions can be demonstrated with the available Help facilities of the MATCH() function.

- Level 1 is the thumb message (Fig. 1). The lookup_value is correct. However, the second argument, the lookup_array is not, since only a one-dimensional array is accepted, so the correct argument should be lookup_vector. The third argument is the match_type, where the “less than” and the “greater than” expression appear, which are not much help. It would be much better to say that 1, !1, or 0 has to be typed when the values of the vector are in ascending order, descending order, or non-ordered, respectively. Then the thumb message is expanded: “Finds the largest value that is less than or equal to lookup_value. Lookup_array must be placed in ascending order.” The message only at the end contains the most important information: “Lookup_array must be placed in ascending order.”, however, no average user gets this far.

- Level 2 is the wizard. At this level, the description of the MATCH() function is the following: “Returns the relative position of an item in an array that matches a specified value in a specified order.” While the definition of the lookup_value: “Lookup_value is the value you use to find the value you want in the array, a number, text, or logical value, or a reference to one of these.” There is no end-user who can understand these. It would be easier to say that the “lookup_value is the value you are looking for”.

- Level 3 is the official help. This is the best, since it gives e.g. the following description: “The MATCH function searches for a specified item in a range of cells, and then returns the relative position of that item in the range.” and the following short definition for the lookup_value “The value that you want to match in lookup_array.”. We have to go the third level to get some valuable information about the MATCH() function. However, no end-user would do this on regular bases.

In the original SCF in the comparison of the lookup functions the following partially correct description can be read: “Approximate matches with VLOOKUP, HLOOKUP and MATCH will only work if the first column of the table being referred to is sorted correctly.”

- In the case of the VLOOKUP() and HLOOKUP() functions only ascending order is allowed, while in the MATCH() function both ascending and descending orders are permitted. This is one of the advantages of the MATCH() function over the VLOOKUP() and HLOOKUP(). Beyond that, all three functions handle non-ordered data.

- For the HLOOKUP() function the first row should be ordered, not the first column. With the MATCH() function it does not matter whether the vector is a row or a column. This is the second advantage of the MATCH() function.

- In the case of the MATCH() function the order of the vector with the matrix does not matter, which is the third advantage of the MATCH() function. Consequently, the “first column/row” restriction is only applies to the VLOOKUP() and HLOOKUP() functions.

Considering all these points, a well-defined algorithm can be formulated to handle vector based items: (1) defining the look-up value, (2) finding the position of the record, and (3)
displaying the selected value of the record in question. This algorithm can be coded extremely conveniently with the `INDEX(MATCH())` composite function. The first step of this algorithm cannot be defined more precisely, since it is highly dependent on the problem to be solved (Csernoch, 2014).

A further advantage of teaching and using both `SEARCH()` and `MATCH()` functions is that this approach provides strong support for knowledge transfer within spreadsheets. The two functions are based on similar algorithms and the output value of both is a position, while the only difference between them is the input data type, with all its consequences.

**Examples of building error-resistant formulas in the SCF**

As is detailed in this paper (Section 3), recognising and handling errors must be a part of the training process from the very beginning. Manual data analysis and recognition of data types would serve novices extremely well. However, general users can handle error outputs with the `ISERROR()` function, and consequently, with the `IF(ISERROR())` composite function. As was mentioned earlier with other groups of functions, there is no need for problem specific functions handling different errors, since this combination would serve all purposes. In this context, where computational thinking is focused on, one further advantage of the simple `ISERROR()` function is that users are “forced” to figure out the source of the error, which is a very similar process to the selection of the appropriate error function and the understanding of its output.

In the SCF error-examples, similar to the previous ones, the focus is again on technical details and version compatibility, instead of the algorithm of handling errors, along with explanatory error messages and their role in programming.

**3. The Edu-Edition of the “Spreadsheet competency framework”**

The following two hypotheses play an extremely important role in our reasoning; spreadsheet competences should be considered (1) at an earlier stage than the “finance professional” and (2) for a much wider range of users. In general, we argue that developing spreadsheet competences should start in and with education. Taking this into consideration, in what follows, we introduce the “Edu-Edition Spreadsheet Competency Framework” (E²SCF).

In the E²SCF we argue that developing spreadsheet competences must be started in formal education as early as possible, focusing on real world problem solving with high mathability approaches (Biró & Csernoch, 2015a, 2015b), which support knowledge transfer and offer a more general computer science approach to spreadsheets.

One of the major points of our argument is that problem solving and knowledge transfer should be focused on when considering digital competences. In this sense, problem solving includes the major steps of real world problem solving: (1) analysing the problem, (2) building algorithms, (3) coding, and (4) discussion, debugging (Pólya, 1954). As was mentioned earlier, problem solving should not be the privilege of highly qualified spreadsheet professionals. It should be part of the training and the application of spreadsheets at any level, starting from basic users. Problem solving and programming should not be mystified (Ben-Ari, 2011). Instead, the difficulty level of problems should be defined and matched with the level of users (Skemp, 1971). Similar to real world problem solving, design, data analysis, and error handling should also be present at any level.

Data analysis, especially unplugged and/or semi-automated data analysis, should be introduced as early as possible. End-users have to be trained to recognize data types, the contents, the possible input and output values, and the connection between them. Without this background knowledge and skill, reliable data management is almost impossible.
In the problem solving approach to end-user training and application schemata-construction also plays a crucial role. This is the key concept for knowledge transfer and reliability. If complete, general schemata are available, the knowledge built up earlier, regardless of the environments, can be transferred and applied in novel situations, and these schemata also lead to a reduction in the number of errors in documents, since fast, intuition-based thinking becomes reliable (Csernoch, 2017; Kahneman, 2011).

3.1. The benefit of spreadsheet knowledge transfer

Knowledge transfer

When working with spreadsheets, a two-directional knowledge transfer is preferable and should be available. Firstly, spreadsheets are open to incoming knowledge and schemata built up in other ICT and traditional environments. Secondly, due to the programming aspect of spreadsheets, knowledge built up in this user-friendly interface can be transferred to more traditional programming environments, to database management, to ICT – data handling and coding, end-user text management, etc. –, and to traditional sciences, primarily mathematics. Finally, due to the simplicity of the spreadsheet environments – considering the data manipulating interface and the toolbars – and the available high mathability spreadsheet-programing approaches and languages (Booth, 1992; Hubwieser, 2004; Wakeling, 2007; Sestoft, 2011; Csernoch, 2014) – considering the algorithm-building and the coding –, even unexperienced debutant and end-user programmers have the opportunity to focus on the problems, the algorithms, and the related discussions, instead of on the coding details.

Spreadsheets would be an ideal introductory problem solving and programming environment if widely accepted institutions and concepts – such as SCF, ECDL, MS, etc. – focused on this aspect, instead of on the tools and the pure usage itself. In the following we outline E²SCF, which strongly supports knowledge transfer with high mathability problem solving approaches.

E²SCF for basic and general users

In E²SCF, first we suggest the modification of the structure of the “framework specification” table. A column should be added to the table which considers the general ICT and traditional subject knowledge brought into spreadsheets – MAthematics (MA), Design and Planning (DP), Incremental Nature of Science (IS) (Chen et al., 2015), authentic contents (AC) –, input knowledge for short (IK). We also suggest a change in the order of the Level 1 items and in several cases changes in the subgroups. Since we focus on general education and the two-directional knowledge transfer, our competency table considers only two levels; basic and general users (BU and GU). The competences listed in the table would serve as the basis for further studies in ICT and also as a tool for strengthening concepts in other sciences, especially in mathematics.

We can read in the SCF that “individuals below the basic user level should not be in a position to access an organisation’s spreadsheets, as they are unlikely to use them safely and effectively.” (SCF, 2016 p. 4) We go one step further and claim that there is a certain ICT background which is required for working safely and effectively in computer related activities. Until this level is reached, individuals should not be allowed to work in digital environments for any company. However, we must also note that working in spreadsheet environments would develop and strengthen users’ ICT knowledge and security. A reversal knowledge transfer is possible if spreadsheets are not taught in isolation but as part of the greater ICT community. Consequently, training must also focus on this aspect of knowledge transfer.
Table 1. The framework specification of E²SCF considering the knowledge brought into spreadsheet management through knowledge transfer and spreadsheet competences of basic and general users (BU and GU, respectively)

<table>
<thead>
<tr>
<th>Item</th>
<th>Input knowledge</th>
<th>BU</th>
<th>GU</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Problem solving</strong></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Breaking down and researching problems</td>
<td>MA, DP, IS, AC</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Tracing errors in spreadsheets they build</td>
<td>MA, ICT, IS, AC</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Building error-resistant formulas</td>
<td>MA, ICT</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Understanding manual vs automatic calculation</td>
<td>ICT, MA, IS, AC</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Recognizing error messages</td>
<td>ICT, MA, IS, AC</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Handling data-entering error messages</td>
<td>ICT, MA, AC</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Handling formula-entering error messages</td>
<td>MA, AC</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Handling data-driven error messages</td>
<td>AC</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Recognizing data types</td>
<td>ICT</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Analysing data manually</td>
<td>ICT, ICT</td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>Basic ICT skills</strong></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Accessing and saving files</td>
<td>ICT</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Reading and entering data</td>
<td>ICT</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Manipulating set up and printing</td>
<td>ICT</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Naming files</td>
<td>ICT</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Converting files with Save As</td>
<td>ICT</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Managing find and replace processes</td>
<td>ICT</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Understanding and applying navigation shortcuts</td>
<td>ICT</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Understanding and applying copy and move shortcuts</td>
<td>ICT</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Understanding and applying file management shortcuts</td>
<td>ICT</td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>Design and best practice</strong></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Designing layout</td>
<td>DP, ICT, AC</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Explaining calculations they build</td>
<td>ICT, MA, AC</td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>Formulas</strong></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Understanding and applying basic arithmetic</td>
<td>MA</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Understanding the concept of functions</td>
<td>MA, ICT</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Calling non-array-based general purpose functions</td>
<td>MA, ICT</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Understanding and handling vectors</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Building vector output array formulas</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Building one value output array formulas</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Calling array-, error-, and condition-based general purpose functions</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Building 2 and 3-level composite functions</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Building multi-level composite functions</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Understanding precedent and dependent cells</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>Formatting</strong></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Understanding and applying hiding, unhiding, deleting, inserting rows, columns, cells</td>
<td>ICT</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Understanding and applying grouping, merging</td>
<td>ICT</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Understanding and applying regular cell formatting</td>
<td>ICT</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
3.2. Tools for managing E^SCF

Array formulas vs. copying and references

Introducing array formulas at the basic level might seem challenging. However, our testing has proved that even beginners welcome the concept, learn it fast, and apply it safely (Biró & Csernoch, 2016a, 2016b; Csernoch & Biró, 2013, 2014). One of the advantages of array formulas (Walkenbach & Wilcox, 2003; Walkenbach & Wilcox, 2003; Walkenbach, 2002, 2010; Csernoch, 2014) is that both copying formulas and using absolute and mixed references are avoidable, which plays a crucial role in spreadsheet security, since these are two of the major sources of spreadsheet errors. Furthermore, using vector-output array formulas, single items cannot be changed, unlike in copying, which makes spreadsheet formulas safer. In educational environments – in a classroom – one further advantage of array formulas is that it makes the modification of the formulas faster, in the sense that teachers do not have to check and warn students repeatedly whether they have modified the formula in the first instance nor remind them about copying.

Beyond this, array formulas strongly support the usage of variable vs. constants and single-value array formulas can substitute various problem specific built-in formulas. The other reason that absolute and mixed references are left out from E^SCF is that experiences prove that the phenomenon of references is one of the most difficult concepts in spreadsheets; consequently, basic and general users are not ready for it. In general, if array formulas are built there is no need for absolute and mixed references, at this level. Considering the handling of array formulas, we have found that it is more natural to use them in tables, with data arranged into fields – vectors – than handling all the cells as individual items. We also tried our method with primary and middle school children and found that with our unplugged tools it is a lot easier to handle a set of data as one object (Biró & Csernoch, 2017). Beyond that, due to the user-friendly environments of spreadsheets, the definition/declaration of a vector is only a selection, which is extremely convenient. We are aware of the debate over the changing of the size of the arrays when we handle vector-outputs. However, there are methods in which the changes in the size can be handled in flexible ways.

One further advantage of defining and using array formulas in spreadsheets is that students can be prepared for programming in imperative languages. The concept of array – vector and matrix – is introduced in an environment where the declaration and definition of array is extremely convenient: only a selection of a range on the graphical interface. Beyond this, the concept of loop is also introduced, since with an array formula repeated activities are carried out on the items of the arrays.

Functions

Research and experience have proved that for beginners no more than 12–15 functions or instructions can be taught and used effectively (Hromkovic, 2009; Walkenbach, 2002, 2010; Walkenbach & Wilcox, 2003; Wilcox & Walkenbach, 2003). Based on these findings, we defined a dozen functions, entitled Sprego functions (Csernoch, 2014), which serve as the introductory set for basic and general users. It has also been experienced that trainees without any special education in mathematics are better at handling text based and text oriented problems, and consequently the functions handling these problems. Considering all these, the set of Sprego functions consists of

- four text functions – LEN(), LEFT(), RIGHT(), SEARCH(),
- four maths functions – SUM(), AVERAGE(), MIN(), MAX(),
- four functions for handling conditions, arrays, and errors – IF(), MATCH(), INDEX(), ISERROR().
According to the problems, the set of Sprego functions can be expanded with further general purpose functions. Our suggestion is the following: SUBSTITUTE(), SMALL(), LARGE(), AND(), OR(), NOT(), INT(), ROUND(), RAND(), OFFSET(), ROW(), COLUMN().

Using only a limited number of functions has the advantage that students can remember them, so knowledge and schemata can be stored in long term memory, and can be called up and activated in problem solving in fast, safe, and effective ways.

One further advantage is that handling other, non-mathematical functions would clarify the concept of function introduced in maths classes. In spreadsheet environments, and especially in Sprego, n-ary and composite functions are introduced, handled, and required for problem solving, something which – currently – is mostly avoided in elementary and high-school mathematics. With opportunities to work with “real” n-ary functions a reversal knowledge transfer is possible, from ICT to mathematics. Handling composite functions in Sprego is similar to traditional programming languages: the decision to build composite functions or use additional variables and/or arrays is always guided and ruled by the requirements and nature of the problem and the programming environment. The creation of the composite functions and embedded structures would also be part of the reversal knowledge transfer. A concept which is hardly mentioned in maths classes, is however, a basic element of knowledge in traditional programming.

Real world problem solving and the levels of understanding in spreadsheets

The E²SCF extensively supports the usage of authentic tables – used in Csernoch, 2014; Csernoch & Biró, 2015a, 2015b and defined in Csernoch & Biró, 2017. In brief, authentic tables contain real data whose content can be selected in accordance with the students’ interest, and as such can be highly motivating (Angeli, 2013; Ainley & Pratt, 2005; Cooper & Dunne, 2000) and easily converted into real world situations. Beyond considering the content, with this method, incoming knowledge in the form of handling files is activated in an intensive way. Consequently, authentic tables can provide data which motivate students to use spreadsheets. Research has clearly proved that one of the reasons for failure when teaching spreadsheets is the decontextualized and technocentric teaching methods (Angeli, 2013; Csernoch & Biró, 2016a, 2016b; Mireault, 2016; Csernoch, 2017), a tendency which is recognizable in the original SCF. Finally, motivation and spreadsheet-supported problem solving also play a crucial role in the acceptance of end-user computing, something which does not happen at present (Panko, 2013, 2015, Panko & Port, 2013, Kadijevich, 2013).

To test and evaluate solutions for real world spreadsheet problems we adapted the SOLO categories of understanding (Biggs & Collis, 1982; Lister et al., 2006), originally set up for programming tasks and problems (Biró & Csernoch, 2014). With this method we are able to follow the students’ selection of functions, their effectiveness, and their understanding of the formula they selected, and the algorithm they built to solve the problem.

Our tests prove that the usage of the problem specific functions is mainly restricted to formulas holding constants and equality, and one function can only serve one specific problem. Beyond this, our tests clearly show that learning problem specific functions with restricted usage does not support either knowledge transfer or schema-construction. Consequently, those students who are grounded in these functions do not have the ability to generalize; they apply low mathability, ineffective, and erroneous solutions, if they apply any at all.
4. Conclusions

The present paper introduces the Edu-Edition of the Spreadsheet Competency Framework (E²SCF). We argue that spreadsheets should be taught and handled as part of the greater ICT world, focusing on two-directional knowledge transfer, data management, the programming aspect of spreadsheets, and computer aided real world problem solving, in general.

Based on the Spreadsheet Competency Framework for finance professionals, we set up the education framework specification for basic and general trainees and users. Compared to the original framework specification, this table contains an additional column which presents the incoming knowledge necessary for faster and more reliable spreadsheet management: ICT bases and problem solving abilities and skills transferred from other sciences and ICT environments. The other major feature of our framework specification is that problem solving is required at any level, starting from novice end-users, which involves real world problem solving based on authentic tables and contents.

Beyond the framework specification table, we provided the essence of Sprego programming which is a supporting tool for E²SCF. With this approach not only can firm schema-based spreadsheet knowledge be built through real world problem solving, but a reversal knowledge transfer is supported, which affects further studies in ICT, especially in programming and data management, and influences other traditional sciences.

6. REFERENCES


The Role of Spreadsheets in Clinical Decision Support: A Survey of the Medical Algorithms Company User Community

Simon Thorne sthorne@cardifmet.ac.uk
Cardiff Metropolitan University, the Medical Algorithms Company

Abstract

This paper presents and discusses the results of a small scoping survey of Clinical Decision Support System (CDSS) users from the Medical Algorithms Company website which hosts 24,000 different CDSS. These results are analysed, discussed, and compared with other similar studies and contribute to the wider understanding of how CDSS impact on clinical practice. The results show that CDSS provided by Medal are being used by clinical professionals in a variety of settings, both as an operational tool and as a research and reference tool. Whilst these tools are implemented and executed in a database, the initial logic is worked out on a spreadsheet. The paper describes that process and examines some of the results of the survey.

1.1 Introduction

This paper considers the Medical Algorithms Company user base and their approaches to making use of the Clinical Decision Support Systems (CDSS) and associated resources offered by Medal. The paper presents and analyses results from a small survey (P=150) conducted with the users that reflects on the impact that such resources have on clinical practice.

1.2 The Medical Algorithms Company (Medal)

The Medical Algorithms Company is an organisation who specialise in converting algorithmic medical research, published in high quality peer reviewed medical journals, to small computer programs to assist in a wide variety of clinical decision making scenarios. The CDSS offered by Medal are CSS files executed in a database, the user can input patient and condition data, calculate an answer, and output a recommendation. Medal has approximately 24,000 different algorithms over a wide range of medical complaints. These range from Body Mass Index (BMI) through to how much anti-venom to administer in the event of a rattlesnake bite. Medal CDSS are accessed either through a web based interface or through an IOS/Android application. Medal facilitates users through a number of options, there are 4500 CDSS that can be accessed publicly either through the website or the cell application. Medal also offer Application Programming Interface (API) access to the full 24,000 CDSS which organisations can integrate into their own infrastructure.
1.3 Medal and spreadsheets

Iyengar [2009] discusses the now historical approach taken by Medal to provide CDSS to the clinical community via spreadsheet models. Medal at that time had 13,500 different CDSS spreadsheets across 45 medical specialties. These spreadsheets could be downloaded and used as needed. The rationale behind this is that spreadsheet technology is pervasive and can be used by practically anyone, hence this allows provision of important medical knowledge to many potential users. It was also thought that users of the spreadsheets could adapt and modify them to their specific needs which could facilitate an ever closer match of resource and need but also presents a potential source of error. Eventually it was judged that spreadsheet models are too open to error, piracy and misinterpretation and hence Medal considered other modes of deployment for CDSS.

As Medal has progressed as an organisation, the approach to providing CDSS has also evolved. Medal no longer provides spreadsheets for download or calculation, instead all CDSS are implemented and executed in a database with validation controls place on input and a better user interface than previous spreadsheet models. CDSS are available to users through a web interface, mobile applications and for large scale processing through API. However, spreadsheets are still at the heart of the development and provision of new CDSS by Medal, spreadsheets are used as a tool for developing the initial model and ensuring that the input output calculations reflect the clinical knowledge accurately [Shiverly and Harl, 2017]

1.4 CDSS development process at Medal

The development of CDSS at Medal is achieved through multiple identification, development and testing steps.

```
Analytic development of scientific evidence

Analytic identification

Porotype spreadsheet model

• Identify relevant clinical research
• Medical guideline produced

Spreadsheet development

• Spreadsheet developed
• Tested against medical guideline
• Tests completed using extreme values and unit testing

Database model development

• Computational logic converted to database model
• Unit tested and extreme value tested
• All test recorded

Clinical review and verification

• Clinical review of the resulting database model to ensure it accurately reflects the medical guideline

Live review

• User acceptance testing
• Users highlight issues if they occur
• Another process of testing is applied and documented
```

The first step of delivering a new CDSS is to identify suitable algorithms published in high quality peer reviewed medical research journals. This process requires a clinical expert to identify potentially interesting algorithms to implement. Once a paper has been identified, the knowledge contained in the paper is separated into ‘chunks’ as described by Simon [1974]. Chunks are a unit of knowledge measurement, Simon [1998] describes that most experts can accommodate between 50,000 to 100,000 chunks of information on a particular subject. Simon [1998] discusses the English language to contextualise chunks:

“English speakers are experts on the English language — we have stored over 100,000 familiar chunks, which are called words. When we see them in a text, we recognize them and retrieve their meanings from memory”

In the case of algorithmic medical research, a single chunk is an equation that calculates part of the outcome. Between 5 and 20 chunks then form a “medical guideline” which is a document that aims to guide a clinician on decision making and best practice on the diagnosis, treatment, prognosis, risk/benefits and management of a particular condition. Medical guidelines are a key tool in clinical practice and are used extensively by clinical professionals across the world. The resulting medical
guideline is abstracted in word, containing references and annotations all based on the style and conventions of Chemical Abstracts.

Once all of the chunks have been identified in the research and the medical guideline has been produced a prototype spreadsheet model of the guideline is created. At this point a clinical expert reviews the Excel model to ensure that it reflects the medical guideline created. The model is then tested by unit testing all input and output paths of the model and by testing extreme values and ranges. The test data and outcomes are stored for future reference. If these tests are passed then the spreadsheet is given to a programmer who implements the computational logic in a database. Once the database implementation is completed the database model is medically verified by the clinical team using the guideline and prototype spreadsheet to cross reference the database. If these tests fail, the process restarts at the medical guideline level.

If these tests pass, the algorithm can then become a deliverable module in the database and is documented. Once enough chunks have been identified and implemented, they are linked together in a modular fashion. Modular development allows reuse of code in algorithms and allows easy changing of components without the need to completely redevelop the entire solution. The clinical knowledge contained in modules is broken down into basic science, diagnosis, therapy, monitoring and patient concerns. Each of these has subtopics. The modules that are built can correspond to these. In theory, you can build a complete process addressing a specific clinical issue like diabetes, osteoarthritis and others. The final part of the process is to consider the language and notation issues from the medical guideline and then the CDSS is delivered.

It is thought that there might be a million chunks of knowledge currently in healthcare although several hundred CDSS would probably satisfy most CDSS users. The use of CDSS allows clinicians to be generalists who can through the use of CDSS knowledge to assess a wider range of clinical problems and therefore reduce the number of referrals needed. For instance a General Practitioner could treat nutrition problems with the right tools rather than referring a patient to another specialist. Therefore, there is significant cost savings that can be gained from using such resources. The process of developing CDSS in this fashion allows minimal costs to the organisation resulting in maximising the number of CDSS that can be provided to the clinical community.

1.5 The Medal user survey

A scoping survey was carried out by Medal with its users to gain more insight into the profile of the users, how the resources are used and what controls and checks are used by individuals and organisations.

1.6 Research questions

The principle aim of the survey was to answer the following research questions:

1. How are Medal CDSS and associated resources used by clinical professionals in terms of: Frequency of use; Experience and training levels with CDSS; Approaches to selection of CDSS; Validation and type of use; Importance of the resources to individuals and organisations and standards imposed on CDSS use by organisations.
2. What specific benefits and disadvantages do users perceive in utilising CDSS in clinical medicine and how do their assessments compare with our understandings from the literature?
3. How does the knowledge contained in CDSS impact on clinical practice and decision making?

However, for the purposes of this paper, not all research questions and data will be explored. This paper shall focus on the issues that are relevant to spreadsheet risk and approaches to managing such risk.
In order to better align this research with spreadsheet risk, the following additional research question is identified.

4. From the Medal user data on spreadsheet based CDSS use, what knowledge from spreadsheet risk management could benefit organisations and individuals making use of spreadsheet based CDSS?

2.0 Computing in Medicine

Computing in medicine has multiple applications, some that are well established such as the use of computers in medical imaging, and others that are still not fully exploited such as Electronic Health Records (EHR) and CDSS. Computing power has the potential to revolutionise medicine in both efficiency and through communication of knowledge Kawamoto et al. [2010b]. Some computer systems are aimed at providing clinical support to medical professionals, some check potential drug-drug interactions, some provide reminders and alerts for clinical workflow and others are designed to manage patient data such as EHR.

2.1 Electronic Healthcare Records

Electronic Healthcare Records (EHRs) are a computer based replacement of the paper charts and records associated with patient care. EHRs offer more a more cohesive and centralised approach to gathering and managing patient data and in theory are more efficient and reliable than paper equivalents. EHRs have the potential to unite a number of disparate computing resources used in medicine, for instance pooling patient data, relevant drug and intervention reminders, clinical workflows and Clinical Decision Support Systems (CDSS) results all at the point of care. However, currently there is little integration of these systems [Kawamoto et al. 2010b] and the use of CDSS in particular remains ad-hoc as a result.

2.2 Clinical Decision Support Systems

Clinical Decision Support Systems (CDSS) are defined as software artefacts produced to assist in clinical decision making scenarios. Clinical decision making scenarios encompass a diverse range of activities that include but are not limited to: Workflow process clinical decision support; Research and reference for particular clinical problems; As an educational tool for learning and teaching medicine; As a tool for evaluating clinical problems by non clinical professionals; As a tool for patients. CDSS users include trained clinical professionals, training clinical professionals, non-clinical professionals in medical industries (such as Electronic Healthcare Records Companies and Medical Insurance) and patients.

2.3 Methodology

This section considers the design of the scoping survey in terms of questionnaire design, sampling, and delivery of the survey.

2.4 Questionnaire design

A questionnaire was chosen as the research instrument for this study since it is ideal at gathering shallow broad information ideal for a scoping survey. The survey serves two distinct purposes, firstly learn more about the Medal userbase and secondly to evaluate the impact of Medal resources on clinical practice.

The questionnaire will examine the user base along the following lines of enquiry: demographics, experience and frequency of use of the resources, approaches to selecting algorithms, validation and
standards, importance of resources (organisational and individual), modes of access to resources, efficiency and specific benefits and limitations to individuals.

2.5 Participant recruitment and sampling

The sampling strategy for this survey is a clustered random approach. The cluster identified were users of the Medal website but no other filtering was applied, hence the random selection of the visitors or members of the site. This approach was chosen since the survey was aimed at discovering more about Medal users and, more broadly, CDSS users in general. Participants were recruited via a message on the splash page of the main Medal website and through an invitation to participate in the survey sent in the weekly email blog. Medal has around 40,000 registered users although it is not known how active those members are. The Medical Algorithms Company’s website receives around 7000 unique hits per month but again little is known about those visitors other than they are unique. Hence the potential population is relatively large, 150 responses were obtained over a period of around 4 months.

3.0 Results

3.1 Demographics

The demographics of the survey show that the large majority of the participants (40) were from the United States. The UK, India and Italy also featured 10 and 9 responses respectively. The remaining 91 responses were distributed amongst 33 other countries demonstrating the international nature of the resources Medal provide. 150 responses were gathered in total.

When asked about their occupation, 78% of the participants stated that their occupation was either Clinician, Healthcare Professional, Surgeon, Medical student or Nurse.

3.2 Experience, Training and frequency of access

This section considers user levels of experience with the CDSS tools, relevant training, the frequency of visits to the Medal site and how many calculators might be accessed in a typical single day.

Figure 6 shows the self perceived levels of experience with the Medal CDSS, as can be observed, over three quarters of participants indicated they had either little or experience with the resources or that they had beginner level experience. This finding is reflected in the literature [Kawamoto et al. 2010b], the provision and uptake of training is a major issue in the widespread adoption of CDSS. Whilst these resources are not spreadsheets as such, they are based on spreadsheet models.
Figure 1 Perceived experience with CDSS

Figure 1 details the training received by participants in the use of CDSS, over three quarters of the respondents either have no training or are self-taught, hence most do not have professional training in use of CDSS. The literature reflects this point as appropriate training provision and uptake is poor amongst clinicians [Kawamoto et al. 2010b]. This also reflects the data in figure 2 which shows that most Medal users consider themselves as either having little or basic levels of experience with the resources.

Figure 2 Training in Medal resources

When asked about frequency of access, 19% of participants access medal resources daily, 37% access the resources on a weekly basis and 20% access once a month. This is interesting since it shows that about 56% of the participants are accessing the Medal resources at least once a week. This shows that the resources are used frequently by the participants which would suggest that the CDSS resources are valuable to the participants either in their own personal work or through their organisations aims.

When asked about how many Medal calculators the participants accessed in a typical day, 30% said that in a typical day there would not access Medal resources. However, 58% of the participants use at least one calculator a day and 15% access 4 or more resources per day. This suggests that the Medal
resources are valuable to the participants and that the penetration of CDSS in various medical settings is becoming more common. In total, 46% of the participants indicated that they worked in a hospital setting, of that 46% only 15% said that they didn’t use the resources, leaving 31% as ‘active users’. Of the 31% active users, 41% said they access at least one Medal calculator a day, 42% accessed 2 a day, 7% accessed 3 a day and the other 10% said they accessed 4 or more a day.

These findings are interesting since it suggests that the Medal CDSS are being used operationally in hospitals to assist in diagnosis or research of conditions. Kawamoto [2005] discusses the lack of inclusion of CDSS in clinical workflow processes as a major barrier to the uptake of such resources, some of the Medal users are getting around this by accessing Medal calculators on mobile devices and desktop computers. Of the 37% who use the resources at least once a day, 67% say they use an Android or IOS cell/Mobile/Smart phone for access, 19% use desktop computers for access, 8% use laptops and 5% use Android or IOS tablets. This data seems to confirm that hospital workers are using mobile devices to access these resources operationally to assist in their day to day duties.
3.3 Validation of CDSS results and organisational standards

Validating findings obtained from CDSS is an important step in the use of CDSS to enhance medical care. Equally, the organisations attitude towards the use of CDSS is another important part of the safe and efficient application of CDSS technology. This coming section explores those issues.

Participants were asked about approach used to validate results obtained from Medal CDSS. The most common approach to validating results is to check CDSS data against published relevant medical literature as answered by 44% of the participants; 28% answered they accept the calculators were correct; 18% said they cross check the results with colleagues and 6% sought the opinion of a superior. This means that of the 150 total participants, 69% employ some form of check to validate the results. The validation process is made easier if the CDSS are part of the clinical workflow and hence standards can be applied to ensure validation and evaluation of results (Kawamoto et al. 2010b).

Two questions asked the participants about the importance of the resources offered by Medal both to the organisation they work for and to themselves as individuals. When asked about the importance to the organisation, the most frequent response was “moderately important” with 40% of participants indicating so; 17% said the resources were “very important”, 3% said “critical” and 26% said the resources were “unimportant”. It would seem that organisations do rely on these resources to some extent 60% of participants said that Medal resources were at least moderately important to the organisation.

When asked about the importance of the resources to individuals, the most frequent response was “moderately important” with 49% of participants indicating so; 22% said “very important”; 15% said “unimportant” and 2% said “critical”. In comparison to responses of organisational importance, the results are very similar but fewer said that the resources were unimportant. Perhaps one explanation for this is that the resources have value to individuals as tools to support their activities but organisations have not yet widely recognised the usefulness and importance of the resources fully. A common problem reported in the literature is the lack of CDSS as part of the workflow process and perhaps these results are indicative of this point [Kawamoto et al. 2005, Kawamoto et al. 2010a, Eichner and Das 2010, Castaneda et al. 2015, Bakken et al., 2007].

When asked about organisational standards that govern the use of Medal resources, see figure 3, 66% said there were no standards set; 16% said there were informal unwritten guidelines, 14% said they have basic written standards and 5% said there were detailed written standards. Consider these results in light of figures 12 and 13 which both show that Medal resources are important to both organisations and individuals. It would seem that organisations have failed to acknowledge and incorporate Medal resources, and other non Medal CDSS, into their workflows and in turn, internal standards for use. The development and deployment of standards associated with CDSS are important since ad-hoc use of such systems lends itself to misunderstandings and sub optimal efficiency. This finding is echoed by Kawamoto et al. [2010b] who discuss the broad lack of standards associated with CDSS use and the lack of formal uptake of CDSS systems in the medical community. In addition, this data suggests that CDSS use is unknown to clinical organisations which is similar in nature to how spreadsheets tend to make up a large part of businesses computing infrastructure without explicit knowledge of the organisation. Panko [2013] described spreadsheets as the dark matter of corporate IT infrastructure, perhaps CDSS will become the dark matter of clinical computing.
When the participants were asked about observation of standards at their organisation, most participants indicated that they do observe standards set, see figure 4. This suggests that if organisations had suitable policies, they would be observed. Hence organisations seem to be trailing behind the medical community who are making use of such resources.

**Figure 3 Standards and Policies that govern CDSS use**

**Figure 4 Observation of standards and policies**

### 3.4 Benefits and Limitations to Medal resources

Figure 5 shows how the participants use the Medal Calculators, 36% indicated they use the calculators as an educational learning and teaching tool; 34% indicated they use the calculators as a research and reference tool; 30% indicated that they use the calculators as a day to day decision support tool and <1% indicated research and development. These results show that Medal calculators are used in two basic modes, either as a research, reference, educational or teaching tool, or they are used operationally to make day-to-day decisions.

Further, if one isolates the 61% of participants who indicated they worked in a hospital or medical practice, 66% say the use as research and reference tool; 63% say they use the calculators as a day to day decision support tool and 73% say they use the calculators as an educational learning or teaching
tool. This suggests that those who work in hospitals or medical practices are using multiple strategies to exploit the resources since several indicated that they use Medal CDSS in all categories of the question posed. Further, since these participants work in either a hospital or general medical practice, one can infer that staff are operationally using Medal CDSS to assist in the diagnosis of conditions, as research and reference material and as an educational learning or teaching tool at the point of care. This provides further evidence that CDSS have a practical and essential operational use.

Figure 5 Approaches to use of Medal CDSS

When participants were asked how Medal calculators impact on work performance: 36% said they can broaden and deepen their knowledge; 33% said they achieve greater time efficiency; 17% said the resources allow them to communicate with patients more efficiently; 12% said the resources allow them to communicate medical conditions to clinical and nursing students, 1% said validation of diagnosis and 1% said more efficient research. See figure 6. Clearly, the Medal calculators offer multiple benefits to the participants as both an operational front line tool but also as a study and reference tool to increase an individual's knowledge and to teach others.

Figure 6 Effect of Medal CDSS on performance
3.5 Discussion

This next section considers the results of the survey and explores how these issues relate to spreadsheet risk management, both in terms of risk presented using these resources but also what can be bi-directionally learnt from CDSS use and spreadsheet risk management.

Although this survey relates to CDSS provided by Medal, it is reasonable to assume that Medal are not the only source of CDSS used by participants and that spreadsheets are probably used in addition to these resources across the clinical community. Croll and Butler [2006] explored clinical spreadsheets publicly available on the internet and found over 800 references to spreadsheet applications by searching the pubmed database (https://www.ncbi.nlm.nih.gov/pubmed/). In addition, although the questions posed relate directly to the Medal CDSS resources, some questions have wider implications such as the development and adoption of organisational standards for CDSS resources.

3.6 Risks and mitigation of CDSS use

From the data it is clear that there are certain risks that can be identified in the practice of individuals and organisations relating to CDSS. There are also some corrective and preventative actions that can be taken to mitigate such risks which will be discussed in turn.

Figure 1 shows that over three quarters of the participants consider themselves to have either little or no experience or beginner level experience. In addition, when asked about training over three quarters said they either have had no specific training for such resources or that they have self-taught themselves to use the resources, see figure 2. This highlights a deficiency in current education practice since such resources are likely to become increasingly important in future medical care. There is also an interesting parallel to be drawn with spreadsheet training. Research in spreadsheet development shows that most users either have no formal training or they have self-taught, Lawson et al. [2006] discuss a survey of 1597 MBA alumni from various HE institutions in the United States and Europe. When asked about training, 40% said that they had no formal training in the use of spreadsheets and that being ‘self-taught’ was the most common response. This issue is echoed in other research too such as [Taylor, 1998] which shows that most ‘end users’ do not have formal training in the design and implementation of spreadsheet tools. This should be rectified by organisations devising and offering clinical professionals training in the use and interpretation of CDSS.

When asked about validation of results obtained from CDSS, 70% employ some form of double checking either through checking the medical literature, seeking the opinion of colleagues or the opinion of a superior. However, 30% said that they accept that the calculators were correct implying that they employ no other form of checking. Whilst medical professionals are highly trained, this presents a risky behaviour amongst the participants since misconceptions, misinterpretations and overconfidence can affect even highly trained and experienced professionals [Lusted 1977, Oskamp 1965] which could lead to poor decision making with significant consequences. Overconfidence is a difficult issue to mitigate since it is so widespread, Panko [2003] discusses approaches to reduce overconfidence in spreadsheet development and finds that informing participants that the spreadsheets produced could contain errors seemed to reduce overconfidence and improve accuracy. In CDSS the same thinking could be applied with the application of checks through a wider set of standards and controls that links CDSS use with EHR.

When the participants were asked about the importance of CDSS resources to both the organisation they work for and to their own personal role in the organisation, participants indicated that CDSS results were moderately important to both their organisation and personal activity. When asked how many CDSS resources they make use of in a typical week, 75% said they use at least one CDSS. When asked about how the resources are used, see Figure 5, 42% said they use these resources to assist in day to day decision making duties. It is therefore reasonable to assume that Medal CDSS are important to organisations and that in the future the importance of these resources is likely to increase. Although CDSS resources have been available for some time, the use of CDSS in medicine is
relatively new phenomena. In time it is likely that such resources will become critical to organisations in the same way the spreadsheet applications are now deemed critical to business decision making. This implies that CDSS use will increase significantly as the technology becomes more common in clinical medicine as will the number of issues regarding use of the technology. Hence clinical organisations should pay special attention to setting standards for CDSS use and formalise the current ad-hoc approach to CDSS deployment, use and integration.

Figure 3 shows that 66% of organisations that participants work for do not have any standards that govern the use of CDSS technology. In combination with the lack of training in such resources, this presents a risk that CDSS could be misused or misinterpreted. This could result in inefficiencies in decision making and potentially more severe consequences through misapplication of knowledge. Clearly this is an organisational management issue that needs to be addressed at the highest levels of the organisations in question to avoid any unwanted consequences of misuse of the technology [Kawamoto et al. 2010b]. Hand in hand with the lack of standards is the current lack of integration of CDSS with other clinical systems such as EHRs, this issue is commonly reported in CDSS literature [Hunt et al. 1998, Kawamoto et al. 2010b, Das and Eichner 2010, Ash et al. 2003, Garg et al. 2005]. Integration is critical since it streamlines the use of CDSS resources in clinical settings and allows organisations to monitor and control the use of CDSS. This first step is critical in both fully exploiting the potential of the resources but also ensuring that the resources are used in a safe and appropriate manner.

For the full potential of CDSS to be realised, organisations must concentrate on integration of CDSS with other clinical systems and develop suitable policies on adoption and use [Moja et al. 2014, Thomas and Coleman 2012]. A fully integrated system could facilitate Evidence Based Medicine (EBM) on a patient by patient basis, providing a dossier of evidence for a decision reached. Through the use of Medal’s API, full integration of CDSS, EHR and EBM is possible however it seems that organisations are not adopting this integration currently. In the pharmaceutical industry, the Food and Drug Administration (FDA) impose strict controls on electronic artefacts used in the analysis of drug trials via title 21 CFR Part 11 [FDA, 2017].

Title 21 CFR Part 11 protects against security violations and ensures the reliability of electronic records. The legislation demands companies provide evidence of: audits, validation, electronic signatures and documentation for any software artefact. The legislation also dictates that electronic artefacts be stored in a secure server so that once the artefact has been created and audited, it cannot be changed without authorisation and access is limited by ‘lock and key’. Authorisation would be needed to implement any changes with subsequent auditing, validation and documentation. Such a system could be adapted for the integration of CDSS as part of EBM. For instance, a CDSS could be used to assess a patient's condition on a particular medical issue, once the calculation is complete, a complete copy of all input and output data would be kept on a secure server. This would greatly improve reliability and accountability and provide an audit trail for the decision reached. Indeed, it is recommended that spreadsheet modelling follows the same approach to reduce fraud and increase accountability [Thorne,

3.7 Conclusions

This paper has described the process of CDSS development at Medal which is achieved via a prototype spreadsheet. Once the spreadsheet has passed a number of tests, the logic is then implemented into a database and is executed via a web interface, mobile application or API access.

The paper then considers the users of Medal CDSS and their attitudes and experiences on a number of important issues. Interestingly, there are many similarities with the spreadsheet modelling world. For instance: training in the use of specific resources is scarce; most organisations do not seem to be aware of extensive CDSS use in their organisations; the level of importance attached to the resources by the users is at least moderately important and organisations do not have any policies that govern their use. Although this survey does not cover individuals creating CDSS, only using, this set of
conditions is rather similar to spreadsheet practices evident in organisations today. They are uncontrolled, ad-hoc, ignored by the management of the organisation and increasingly critical to users.
References


Oskamp, [1965], ‘overconfidence in case-study judgements’, *The journal of consulting psychology*, 29, pp 261- 265

Shiverly, J and Harl, J, [2017], CDSS development processes at Medal, *personal communications with the author*

Simon, H, [1974], How big is a chunk?, *Science*, American Association for the Advancement of Science


This volume contains the proceedings of the eighteenth EuSpRIG conference on ‘Spreadsheet Risk Management’ held in July 2017 at the Imperial Data Science Institute, Blackett Laboratory, Imperial College, London, England.

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.

Front cover images courtesy of EuSpRIG Authors.

ISBN          978-1-905404-54-4