21st EuSpRIG Annual Conference
“The Spreadsheet Crisis: Regaining Control”

= TimingBlock\( \lambda(\text{modelStart}, \text{modelDuration}) \)

= IncomeStatement\( \lambda(\text{unitVolume}, \text{unitPrice}, \text{unitDirectCosts}, \text{depreciationMonthly}, \text{interest}) \)

= DebtSchedule\( \lambda(\text{modelStart}, \text{modelDuration}, \text{debtAmortisationDate}, \text{debtFacilityA}, \text{debtAmortisationAmount}, \text{interestRate} / 12) \)

= WorkingCapital\( \lambda(\text{revenue}, \text{cashReceiptTiming}) \)

= WorkingCapital\( \lambda(-\text{costs}, \text{cashPaymentTiming}) \)

= CashFlow\( \lambda(\text{modelDuration}, \text{openingAccReceivable}, \text{openingCashReceiptTiming}, \text{AccountsReceivable#}, \text{openingAccountsPayable}, \text{openingCashPaymentTiming}, \text{AccountsPayable#}, \text{interest}, \text{repayment}, \text{openingCash}) \)

The Foundling Museum, 40 Brunswick Square, London.
Thursday 6th July 2023
European Spreadsheet Risks Interest Group

EuSpRIG 2023 Conference

The Spreadsheet Crisis: Regaining Control

Conference Supported By:
ICAEW
Cardiff Metropolitan University
Spreadsheet Engineering
Systems Modelling
Operis
Dow Schofield Watts

Proceedings Editor:

Dr Simon Thorne

6th July 2023

The Foundling Museum, 40 Brunswick Square, London.
EuSpRIG 2023 Conference

“The Spreadsheet Crisis: Regaining Control”

EUSPRIG 2023 ORGANISING COMMITTEE

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

EUSPRIG 2023 PROGRAMME COMMITTEE

Dr 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 2023
Conference Programme

08:30 – 09:30 Arrival, Registration & Coffee

09:30 – 10:15 Session 1 (Keynote)
Bringing generative AI to the Excel grid: from research to practice
Andrew D. Gordon and Jack Williams, Calc Intelligence Microsoft Research

10.15 – 11.00 Session 2
Reducing Errors in Excel Models with Component-Based Software Engineering
Craig Hatmaker

11.00 – 11:30 Coffee Break

11.30 – 12:15 Session 3
A Use Case-Engineering Resources Taxonomy for Analytical Spreadsheet Models
Tom Grossman and Vijay Mehrotra, University of San Francisco

12.15 – 13:00 Session 4
How Beaufort, Neumann, and Gates met? Subject integration with spreadsheeting
Maria Csernoch and Julia Csernoch, University of Debrecen, Faculty of Informatics

13.00 – 14:00 Lunch

14:00 – 14:45 Session 5
Excel as a Turing-complete Functional Programming Environment
Peter Bartholomew, MDAO Technologies Ltd

14:45 – 15:30 Session 6
ChatGPT and Excel – trust, but verify
Patrick O'Beirne, Systems Modelling Ltd

15:30 – 16:15 Session 7
Experimenting with ChatGPT for Spreadsheet Formulae Generation: The Risks of AI Generated Spreadsheets
Simon Thorne, Cardiff Metropolitan University

16:15 – 17:00 Tea Break with Question and Answer session

17:00 - Conference Close
AGM to discuss future direction of EuSpRIG.
## EuSpRIG 2023 PROCEEDINGS

### TABLE OF CONTENTS

<table>
<thead>
<tr>
<th>Paper</th>
<th>Author(s)</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>A Use Case-Engineering Resources Taxonomy for Analytical Spreadsheet Models</td>
<td>Tom Grossman, Vijay Mehrotra</td>
<td>1</td>
</tr>
<tr>
<td>How Beaufort, Neumann, and Gates met? Subject integration with spreadsheeting</td>
<td>Maria Csernoch, Julia Csernoch</td>
<td>15</td>
</tr>
<tr>
<td>Reducing Errors in Excel Models with Component-Based Software Engineering</td>
<td>Craig Hatmaker</td>
<td>33</td>
</tr>
<tr>
<td>Excel as a Turing-complete Functional Programming Environment</td>
<td>Peter Bartholomew</td>
<td>59</td>
</tr>
<tr>
<td>ChatGPT and Excel – trust, but verify</td>
<td>Patrick O'Beirne</td>
<td>73</td>
</tr>
<tr>
<td>Experimenting with ChatGPT3 for Spreadsheet Formulae Generation: The Risks of AI Generated Spreadsheets</td>
<td>Simon Thorne</td>
<td>89</td>
</tr>
</tbody>
</table>
PREFACE

Dear Colleagues,

You are very welcome to the 2023 Annual Conference of the European Spreadsheet Risks Interest Group. It was a long wait, four years, since our last in-person conference. We are fortunate to have our unique combination of real-world experience and academic research which is the distinguishing feature of our event.

Our academic track features two papers: “A Use Case-Engineering Resources Taxonomy for Analytical Spreadsheet Models”, by Tom Grossman and Vijay Mehrotra; and How Beaufort, Neumann, and Gates met? Subject integration with spreadsheeting” by Maria Csernoch and Julia Csernoch.

The technical track has a deep dive into Excel’s newest features: “Reducing Errors in Excel Models with Component-Based Software Engineering” by Craig Hatmaker; and “Excel as a Turing-complete Functional Programming Environment” by Peter Bartholomew.

Our core focus on spreadsheet risk considers new risks arising from the use of Large Language Models and chatbots: “ChatGPT and Excel – trust, but verify” by Patrick O’Beirne; and “Experimenting with ChatGPT3 for Spreadsheet Formulae Generation: The Risks of AI Generated Spreadsheets” by Simon Thorne.

These topics will demand your attention and stimulate debate that will go on both in informal meetings afterwards, and in our online discussion forum on groups.io which has over 180 members. Currently we have about 1000 unique visitors to our website per month.

It is my pleasure to once again acknowledge the keen work of our conference and programme organiser Dr Simon Thorne from Cardiff Met. The committee also depends upon the wise counsel and active support of David Colver of Operis, Grenville Croll, and Angela Collins our secretary. Their 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 2023

http://www.eusprig.org

https://groups.io/g/eusprig
A Use Case-Engineering Resources Taxonomy for Analytical Spreadsheet Models

Thomas A. Grossman and Vijay Mehrotra
University of San Francisco, 2130 Fulton St., San Francisco CA 94117 USA
tagrossman@usfca.edu, vmehrotra@usfca.edu

ABSTRACT

This paper presents a taxonomy for analytical spreadsheet models. It considers both the use case that a spreadsheet is meant to serve, and the engineering resources devoted to its development. We extend a previous three-type taxonomy, to identify nine types of spreadsheet models, that encompass the many analytical spreadsheet models seen in the literature. We connect disparate research literature to distinguish between an “analytical solution” and an “industrial-quality analytical spreadsheet model”. We explore the nature of each of the nine types, propose definitions for some, relate them to the literature, and hypothesize on how they might arise. The taxonomy aids in identifying where various spreadsheet development guidelines are most useful, provides a lens for viewing spreadsheet errors and risk, and offers a structure for understanding how spreadsheets change over time. This taxonomy opens the door to many interesting research questions, including refinements to itself.

1. INTRODUCTION

It is well-understood that spreadsheets are indispensable to business (e.g., Croll 2007, Grossman, Mehrotra, and Özlik 2007), but can also be a source of risk and costly errors (e.g., EuSpRIG 2023, Panko 1998, but also Powell, Baker, and Lawson 2008a, 2008b, 2009a, 2009b). Researchers and practitioners have long been working on articulating standards and practices to reduce risk and error, and also to increase the productivity of spreadsheet programmers, and the effectiveness of spreadsheet users in organizations.

Because of the scope and complexity of the spreadsheet space, any professional practice recommendations must be limited to a well-defined “domain” or class of spreadsheets (Grossman, Mehrotra, and Sander 2011, henceforth referred to as “GMS”). This leads naturally into the need for a taxonomy (set of named classifications) for spreadsheets. GMS consider the domain of analytical spreadsheet models, and provide a taxonomy based on how the spreadsheet is used.

![Figure 1: The Analytical Spreadsheet Model Domain (Grossman, Mehrotra and Sander 2011)](image-url)
The domain of analytical spreadsheet models (see Figure 1) is distinct from spreadsheet models in general, and from spreadsheets in general. GMS provide a five-part definition of an analytical spreadsheet model as a spreadsheet computer program, that implements a mathematical model, for purposes of analysis, that serves as an organizational asset, employed in an important context.

GMS indicate that analytical spreadsheet models are primarily about the business logic embedded in the model, and that models that are primarily data-driven with a large dataset and relatively few cell formulas do not fall in this domain.

GMS differentiate among three types of analytical spreadsheet model, namely “personal productivity tool” (something “quick and dirty” and relatively unimportant); “analytical spreadsheet model” (which is used regularly by someone other than the developer), and “spreadsheet application” (that is deployed to multiple less-sophisticated users). They briefly suggest that the recommended amount of spreadsheet engineering effort is higher for each type of respective spreadsheet. GMS then move on and focus on the characteristics and guidelines for “high quality” analytical spreadsheet models, and do not develop the taxonomy.

This paper extends, refines, and formalizes the three-element taxonomy of Grossman, Mehrotra, and Sander 2011 for analytical spreadsheet models into a richer nine-element model that provides significant insight into how we can think about analytical spreadsheet models, how they are created and should be created, and how they are changed over time.

1.1 Contribution

This paper presents a new framework, or “taxonomy” for understanding analytical spreadsheet models. It examines the relationship between the use case that a spreadsheet is meant to serve, and the engineering resources devoted to the spreadsheet’s development. The taxonomy identifies and proposes a name for nine types of analytical spreadsheet model, extending and refining a previous three-type taxonomy. We identify a new distinction between two previously-merged types of spreadsheets, which are the “analytical solution” and the “industrial-quality analytical spreadsheet model”.

We explore the nature and knowledge of each spreadsheet type, including criteria for a spreadsheet model being considered that type, the appearance of that type in the literature, and how that type of spreadsheets might arise in practice. This taxonomy allows us to examine with greater insight (and even with sympathy) the phenomena of both under- and over-engineered spreadsheets and the costs and risks associated with them.

We believe that spreadsheet development guidelines are not “one size fits all” but should instead vary depending on the business context. This taxonomy provides a framework to understand existing development recommendations, and provides a map to spreadsheet types where development recommendations or risk management measures might usefully be articulated.

We anticipate that this taxonomy can help us understand the evolution and risk of an important analytical spreadsheet model, by providing a structure to explain how a spreadsheet’s use changes over time, with or without adequate planning and investment to support that change.

We emphasize that this research is preliminary. We anticipate that there will be refinements and alterations to various aspects of the taxonomy. Much work remains to be done to fully flesh out the details and criteria for each type of spreadsheet, and to test various insights and hypotheses that arise.
1.2 Overview
In Section 2 we critically examine and refine the GMS three-type taxonomy for analytical spreadsheet models, and clear up some confusing elements from the original. We refine and extend it to two dimensions, and place the original three types into the larger two-dimensional structure. The new taxonomy considers the intersection of three use cases, discussed in Section 3, and three engineering resource levels (Section 4). In Section 5, we combine the use cases and engineering resource levels to yield a new two-dimensional taxonomy featuring nine types of analytical spreadsheet model, which we name and explore in detail. We conclude in Section 6 with suggestions for further research.

2. REFINING AND EXTENDING THE GMS THREE-TYPE TAXONOMY
In this section we examine the GMS three-type taxonomy. We critique this taxonomy, clarify and refine it, and extend it from one dimension to two. This prepares us to develop the new use case-engineering resources taxonomy in Sections 3, 4, and 5.

The GMS taxonomy for analytical spreadsheet models, which admits three types of spreadsheet model, is formalized in Figure 2.

![Figure 2: Three-Type Taxonomy (Adapted from Grossman, Mehrotra, and Sander 2011)](image)

The GMS taxonomy considers three types of spreadsheet, the Personal Productivity Tool, Analytical Spreadsheet Model, and Spreadsheet Application. We begin by clarifying the terminology.

2.1 Clarifying Terminology and Revisiting a Key Assumption
In the GMS paper, the term “analytical spreadsheet model” is used in three different ways: (A) the domain of spreadsheet model under consideration (see Figure 1); (B) as the classification of spreadsheet model (i.e., the horizontal axis in Figure 2); and (C) as the name of a cell in Figure 2 (“Analytical Spreadsheet Model (ASM)”).

In this paper, we use “analytical spreadsheet model” only for (A), the domain. (The domain of analytical spreadsheet model is defined in Section 1.) This paper (like GMS) considers only spreadsheets that are analytical spreadsheet models. We replace the other uses of “Analytical Spreadsheet Model” with more appropriate terminology in Section 2.2.

Next, we relax a key assumption in GMS. GMS conflate the level of “spreadsheet engineering investment” and the Type of Analytical Spreadsheet Model. They briefly indicate that personal productivity tools should receive lesser spreadsheet engineering investment, analytical spreadsheet models more investment, and spreadsheet applications even more investment. That is, GMS assume alignment between type of model and the investment to create that model.

However, although real-world development projects (including building spreadsheets) sometimes do receive the “just right” level of investment, they can also be provided with insufficient or even excess investment. (cf. Southey 1837). Therefore, we extend in Figure 3 the taxonomy of Figure 2 to explicitly represent varying investment levels.
The three spreadsheet model types from Figure 2 appear in Figure 3, showing the GMS alignment between spreadsheet engineering investment and type of analytical spreadsheet model.

2.2 Refined Terminology: Use Case and Engineering Resource Level

Next, we refine the axes of Figure 3. We rename the horizontal axis from “Type of Analytical Spreadsheet Model” to “Use Case” which is more descriptive. We use Google’s Oxford Languages definition of use case to mean “a specific situation in which a product or service could potentially be used”. We define three use cases, called “Unique Analysis”, “Business Process-Embedded”, and “Spreadsheet Application”. (These three use cases are discussed in detail in Section 3.)

We rename the vertical axis from “Spreadsheet Engineering Investment” to “Engineering Resources” which is shorter and encompasses resources such as time and managerial attention in addition to expense. We indicate three levels of engineering resources, called “Low”, “Medium”, and “High”. (These three levels of engineering resources are discussed in detail in Section 4.)

This yields the Use Case-Engineering Resources Taxonomy (Figure 4). This taxonomy allows us to consider the interaction between the engineering resources deployed to build an analytical spreadsheet model, and the situation in which that model will be used. We add gridlines to highlight that there are nine combinations of Engineering Resources and Use Case.

Let us now revisit and refine the three types of spreadsheet from GMS in Figures 2 and 3. We rename them, and place them into Figure 5 in boxes that indicate alignment between Engineering Resources and Use Case. We adjust the name of the bottom left box to
“Personal Productivity Spreadsheet” (PPS) rather than “Personal Productivity Tool” because “tool” is too restrictive, and because “tool” can be assigned a different meaning (e.g. Olavson and Fry 2008). The center box is called “Industrial-Quality Analytical Spreadsheet Model” (IQASM) to emphasize that it has received a goodly level of investment. The top right box is called “Planned Application” to emphasize that an appropriate level of engineering resources are used in its development. These refinements are shown in Figure 5.

Figure 5: The Use Case-Engineering Resources Taxonomy Showing the Three Types of Spreadsheet Models That Have Alignment Between Use Case and Engineering Resources

2.3 Alignment, Waste, and Danger?

It is tempting (and implicit in GMS) to assume that Engineering Resources should be aligned with Use Case. If this is true, then our taxonomy boils down to Figure 6, where situations with Engineering Resources higher than necessary represent waste, and situations with Engineering Resources lower than necessary represent danger.

Figure 6: Simplistic Use of the Taxonomy that Assumes Alignment Between Engineering Resources and Use Case is Always Appropriate

However, upon further examination in Section 5, we shall see that this is not the case, and that it can be appropriate for a spreadsheet to reside in a misaligned cell (off the up-right diagonal). But first, let us further develop the taxonomy.

3. THE THREE USE CASES

We now provide more detail on the use-case dimension of the taxonomy in Figure 5. There are three use cases: Unique Analysis, Business-Process Enabled, and Spreadsheet Application. (Disclaimer: It is our hope that these three use cases and aspects encapsulate the bulk of important real-world spreadsheet models – and therefore are useful – but we are not quite prepared to claim that they capture every possible circumstance.) Examples
of spreadsheets from each use case can be found in Grossman, Mehrotra, and Özlük (2007) and elsewhere in the literature.

The **Unique Analysis** use case is where a spreadsheet model is built to undertake a single calculation or to provide a model used for analysis in a situation that is limited in scope or time. The spreadsheet is not shared (expect perhaps within a “tight team” that collaborates well and respects spreadsheet integrity). For example, a financial model for a particular project, or the author’s mortgage interest rate spreadsheet.

The **Business-Process Enabled** use case is where a spreadsheet model is used on a regular or routine basis as part of an established business process. Typically, the model is transferred to and used by the business process owner (and is unlikely to be relevant to anybody else). The nature of these spreadsheets is that they are necessarily further transferred over time when the user changes jobs. For example, a spreadsheet to determine weekly orders from multiple vendors; a monthly budget spreadsheet; a spreadsheet that summarizes cumulative capital investment. This use case seems to be the source of the “Accidental Legacy Systems” observed by Grossman, Mehrotra, and Özlük (2007).

The **Spreadsheet Application** use case is where a spreadsheet model is written by a developer and distributed to other people (with an unspecified level of intentionality and planning), resulting in heavy usage of the spreadsheet by multiple users.

In Table 1, we describe four aspects of each use case: intended frequency of use; who can access the spreadsheet; number of users; and reliability of user behavior.

<table>
<thead>
<tr>
<th>Intended Frequency of Use</th>
<th>Unique Analysis</th>
<th>Business Process-Embedded</th>
<th>Spreadsheet Application</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>One-Off (One project/engagement)</td>
<td>Regular (Routine)</td>
<td>Heavy (Massive)</td>
</tr>
<tr>
<td>Who Can Access</td>
<td>Developer (Tight team)</td>
<td>Select personnel (Transfer over time)</td>
<td>Anybody (The great unwashed)</td>
</tr>
<tr>
<td>Number of Users</td>
<td>One (Tight team)</td>
<td>Moderate (Grows over time)</td>
<td>Widely distributed</td>
</tr>
<tr>
<td>Reliability of User Behavior</td>
<td>Excellent (Not shared)</td>
<td>High (Only bus process owners)</td>
<td>Minimal (Out in the Wild)</td>
</tr>
</tbody>
</table>

*Table 1: Aspects of the Use Cases*

Each column of Table 1 indicates the typical characteristics of spreadsheets for that use case. Not all spreadsheets for a use case will have all these characteristics, but they seem generally to apply. It is important to note that a given spreadsheet could appear in any of the three columns of Table 1, because use case is about the business situation, not about the spreadsheet artifact.

The alert reader will notice that risk of misuse or inadvertent damage to the source code of the spreadsheet is not included in Table 1. This is because risk is affected by the level of engineering resources (Section 4) devoted to spreadsheet development. (This is implicit in the GMS assumption that risk is appropriately managed when there is alignment between Use Case and Engineering Resources.)

The use case aspects in Table 1 will be important in Section 5 when we combine them with the engineering resource levels. We next detail the three engineering resource levels.

### 4. THE THREE ENGINEERING RESOURCE LEVELS

We now provide more detail on the engineering resources dimension of the Use Case-Engineering Resources Taxonomy in Figure 5. There are three engineering resource levels: Low, Medium, and High. We address each in turn, considering several aspects of
how a spreadsheet model is resourced during development. (Disclaimer: It is our hope that these engineering resource levels and aspects encapsulate a significant fraction of important real-world spreadsheet models – and therefore are useful – but we are not quite prepared to claim that they capture every possible circumstance.)

The Low engineering resources level is a situation where relatively little expense and effort is devoted to the spreadsheet, and development is relatively quick. The spreadsheet is developed with little conscious regard for following any development process or meeting any standards. For example, the spreadsheet the author developed to request funding for a trip to the EuSpRIG conference; a back-of-the-envelope estimate of the cost of a new product; the amount of tax-deductible interest associated with a bond that might be issued.

The Medium engineering resources level is a situation where relatively moderate cost, effort and time are devoted to the spreadsheet. There is some attention to process, and some level of development standards (perhaps rigorous and/or documented). To the extent that there are development standards, they are meant to ensure accuracy and reusability by trained personnel, but would be risky for extensive distribution. There is an expectation that the developer has an appropriate level of skill.

The High engineering resources level is a situation where relatively sizable cost, effort and time are devoted to the spreadsheet. These very high development standards build on the Medium engineering resources level, with additional work such as a sophisticated user interface that safeguards source code, vets model inputs, ensures easy future usage, etc. There is an expectation that the developer has an appropriate level of skill (extensive effort by a naïve developer is not considered High engineering resources).

Although these definitions are necessarily rough, we find them to be useful and welcome future research to refine them.

In Table 2, we describe three aspects of each engineering resource level: development standards; cost/effort; and relative time.

<table>
<thead>
<tr>
<th>Engineering Resource Level</th>
<th>Development Standards</th>
<th>Cost/effort</th>
<th>Relative Time</th>
</tr>
</thead>
<tbody>
<tr>
<td>High</td>
<td>&quot;Unbreakable&quot;</td>
<td>High</td>
<td>Lots</td>
</tr>
<tr>
<td>Medium</td>
<td>Formal (Rigorous, Documented)</td>
<td>Moderate</td>
<td>Some</td>
</tr>
<tr>
<td>Low</td>
<td>Informal (Non-Existent)</td>
<td>Low</td>
<td>Not much (Now)</td>
</tr>
</tbody>
</table>

Table 2: Aspects of the Engineering Resource Levels

In the next section we combine the engineering resource level aspects in Table 2 with the use case aspects in Table 1. This allows us to examine each of the nine cells in the taxonomy.

5. THE NINE TYPES OF SPREADSHEETS IN THE USE CASE-ENGINEERING RESOURCES TAXONOMY

This section presents the use case-resources taxonomy. There are three Use Cases, and three Resource levels, yielding a combination of nine different spreadsheet types. These are summarized in the Taxonomy shown in Figure 7. The four shaded cells in Figure 7 are situations where spreadsheet models are commonly seen in the literature and widely accepted as being desirable. The unshaded cells are situations where the literature is less informative – perhaps because these situations are less desirable – but we believe that examples of these can be found in the field.
The value of this taxonomy is three-fold. First, each cell merits its own set of development goals and guidelines. Second, the risk of error and any concomitant business loss is likely to be different for each cell; to the extent that any of the cells have not been adequately studied, this framework provides fertile ground for future research into spreadsheet development guidelines, risk assessment, and risk management. Third, the taxonomy provides a structure (see Section 6.1) for understanding how a spreadsheet can transition (with purposeful intent) or evolve (organically or by accident) over time.

We now examine each of the nine cells. We explain our proposed name for the type of spreadsheet in the cell. We discuss selected literature, and for the unshaded cells provide hypotheses for good and bad reasons why such spreadsheets might arise in practice. Testing these hypotheses (or developing additional hypotheses) could be a fruitful area of future research.

5.1 The Business Process-Enabled Use Case (Center Column)

Recall from Section 3 that the business-process enabled use case is where a spreadsheet model is used on a regular or routine basis as part of an established business process. We discuss the three types of models in this middle column of Figure 7.

**Industrial-Quality Analytical Spreadsheet Model (“IQASM”)**

We start our tour of the nine types of analytical spreadsheet models in the very center cell of Figure 7. This cell combines the *Business Process-Enabled Use Case*, and has *Medium Engineering Resources*; engineering resources are in alignment with the use case. This cell is the home of the **Industrial-Quality Analytical Spreadsheet Model** (IQASM in Figure 7), which is a spreadsheet that satisfies six characteristics identified in GMS: suitable for efficient analysis; readable; transferable; accurate; reusable; modifiable. Notice that these characteristics are exactly aligned with the *Business Process-Enabled Use Case* attributes in Table 1. Implicit in these characteristics is *Medium Engineering Resources* (enough to satisfy the six characteristics, but less than the High Engineering Resources necessary to build an advanced user interface).

Olavson and Fry (2008) refer to spreadsheets in this cell as a “Tool”, and provide a definition consistent with the six characteristics. They indicate that their group has built and handed-off many such spreadsheets. Read and Batson (1999) seem to include IQASMs as they mention (but do not emphasize) hand-over from developer to user. Based on Grossman, Mehrotra, and Özlük (2007), we believe that IQASMs are easily found in the field.
Gold-Plated Spreadsheet Model (“Gold-Plated”)
Moving up, the top-center cell of Figure 7 combines the Business Process-Enabled Use Case and High Engineering Resources; engineering resources may be excessive for the use case. We call this situation a Gold-Plated Spreadsheet Model (“Gold-Plated” in Figure 7), which has development effort that is more than is required to fulfill its job of enabling a business process with a vetted user.

However, creating such a spreadsheet could be desirable if there was a worry that future users would not be carefully selected, or that careful hand-offs might not be sustainable; equally, such spreadsheets could represent a waste of time and money spent on overengineering unnecessary features.

Field Expedient Spreadsheet Model (“Field Expedient”)
Moving down the center column to the bottom cell of Figure 7, we encounter the Business Process-Enabled Use Case and Low Engineering Resources; engineering resources are insufficient for the use case. Here resides what we call the Field Expedient Spreadsheet Model (“Field Expedient” in Figure 7), which is a spreadsheet that is in regular use in a business process but has not been developed sufficiently to make it an IQASM that is most suitable for that role. (We are not sure that “field expedient” is the best name, but it has the benefit of positivity.)

Having such a spreadsheet could be a smart business move, for example a business situation where a fast handoff of a useful but lesser-developed model (accepting that its use might be problematic since it is designed for use only by the developer) is deemed preferable to not having a model at all – that is, it’s a risk, but a smart risk. Equally, it could arise due to thoughtless, naïve, or careless practices. We hypothesize that some number of spreadsheet errors and problems can be traced to Field Expedient spreadsheet models.

5.2 The Unique Analysis Use Case (Left Column)
Recall from Section 3 that the unique analysis use case is where a spreadsheet model is built to undertake a single calculation or to provide a model used for analysis in a situation that is limited in scope or time. We discuss the three types of models in this left-hand column of Figure 7.

Personal Productivity Spreadsheet Model (“Pers Prod”)
A Personal Productivity Spreadsheet Model (“Pers Prod” in the bottom left cell of Figure 7) is the common situation of a minor spreadsheet, which might be the proverbial “quick and dirty” spreadsheet model, which is intended to be used once for something of interest (Unique Analysis Use Case), that gets little developmental time and attention (Low Engineering Resources).

We hypothesize that the overwhelming majority of the world’s analytical spreadsheet models fall into this category.

Analytical Solution Spreadsheet Model (“Analytical Solution”)
Moving up the Unique Analysis Use Case column of Figure 7, we encounter the center row, Medium Engineering Resources. We refer to this cell as the Analytical Solution Spreadsheet Model (“Analytical Solution” in Figure 7), a term originated by Olavson and Fry (2008). This important type of spreadsheet model cannot be resolved in the three-type GMS taxonomy.

We suggest that a reasonable criteria for inclusion in this cell is a subset of the six characteristics for an Industrial Quality Analytical Spreadsheet Model (see section 5.1
above). The use case is one-off analysis by the spreadsheet developer (or tight team), hence we remove the characteristics associated with handing off the spreadsheet to a less sophisticated business process owner, and can skip the investment for the spreadsheet to be transferable, modifiable, and readable (in the sense of easily readable by a person not familiar with the spreadsheet’s development; see GMS). This yields three characteristics that might define the Analytical Solution: suitable for efficient analysis; accurate; reusable. (We revisit this in Section 6 below.)

The Analytical Solution type is vitally important in certain business contexts. Olavson and Fry view them as central to their practice at Hewlett-Packard. The work in section 2.1 of Tennent and Friend (2005) is an Analytical Solution. Read and Batson (1999) seem to focus on Analytical Solutions, but there work is somewhat casual with regard to use case. The work of Swan (2008), and some or all of the models made by spreadsheet manufactories (discussed further in Section 6) seem to be Analytical Solutions (see Grossman and Özlük 2010).

**Hobby Spreadsheet Model (“Hobby Model”)**

The top row of the Unique Analysis Use Case column of Figure 7 embraces High Engineering Resources. We refer to this cell is as the Hobby Spreadsheet Model (“Hobby Model” in Figure 7). These are spreadsheets that are suitable for analytical use by their developer, that have received further investment to make them difficult to damage or use improperly. Since they are being used by their creator, such investment seems excessive, leading to the “hobby” moniker. However, there may be valid reasons for this investment, such as making provision for possible future unplanned evolution to the business process-enabled use case in Figure 7. (Such evolutions are discussed in Section 6.2.)

**5.3 The Spreadsheet Application Use Case**

Recall from Section 3 that the spreadsheet application use case is where a spreadsheet model is written by a developer and distributed to other people (with an unspecified level of intentionality and planning), resulting in usage of the spreadsheet by multiple users, with the possibility of user modifications of the spreadsheet if that is not prevented by application of engineering resources. We discuss the three types of models in this right-hand column of Figure 7.

**Planned Application Spreadsheet Model (“Planned App”)**

The top row of the Spreadsheet Application Use Case column of Figure 7 meets High Engineering Resources. We refer to this cell is as the Planned Application Spreadsheet Model (“Planned App” in Figure 7). This cell contains spreadsheet models that are purposefully built for distribution to distributed users.

One would expect spreadsheets of this type to have sufficient engineering investment so that it is reasonable to send a copy to relatively unreliable personnel, including provisions to prevent users from editing the source code, and trapping input values that will cause errors or spurious results. Several examples of planned application spreadsheet models can be found in Grossman, Mehrotra, and Özlük (2007), hence it is likely that these are easily found in the field.

**Incidental Application Spreadsheet Model (“Incidental App”)**

The Incidental Application Spreadsheet Model (“Incidental App” in Figure 7) arises when a spreadsheet that serves the Spreadsheet Application Use Case and has received only Medium Engineering Resources. Although the engineering resources are comparable to those for the IQASM cell (to the left in Figure 7), these spreadsheets tend to lack provision for source code protection and input error trapping, and therefore carry a certain level of risk when used by unreliable users.
We are not sure what the literature has to say about these. We have heard anecdotes of such spreadsheets rapidly forking source code (making them difficult or impossible to maintain) and generating spurious results that are blamed on the original developer. Such spreadsheets might arise through naivete, for example, sending an IQASM to many people. But it is not unlikely that there are times in a business setting where the benefits of distribution outweigh the expense and risk.

**Accidental Application Spreadsheet Model (“Accidental App”)**

The Accidental Application Spreadsheet Model (“Accidental App” in Figure 7) arises when a spreadsheet serving the *Spreadsheet Application Use Case* has received only **Low Engineering Resources**. These spreadsheets, which have little or nothing in the way of thoughtful construction or conformance to any standards, and are suitable only for use by the developer, seem (we are being polite) poised to generate exciting results as multiple unreliable users have adventures with them.

The caveats that apply to Incidental Application Spreadsheet Models above apply even more strongly here. We are optimistic that a determined researcher could find a reason where such a spreadsheet type is desirable, but we have been unable to articulate any. This class of spreadsheets seems to carry with it high risk of error, including inadvertent or incorrect changes to source code, as well as inappropriate usage. We are not aware of any literature on this class of spreadsheet, but they undoubtedly exist in the wild.

### 6. CONCLUSIONS AND AREAS FOR FUTURE RESEARCH

We present a new taxonomy for understanding analytical spreadsheet models, which are a domain of spreadsheets of particular practical importance. Nine types of spreadsheets are identified, whereas the previous GMS taxonomy had only three.

The use case-engineering resources taxonomy is grounded in three use cases: spreadsheet models meant for unique analysis; business-processed enabled spreadsheet models that are embedded in a regular or routine activity; and spreadsheet applications that are handed off to multitudinous users. The taxonomy incorporates three level of spreadsheet engineering resources applied to models for each use case: low resources where little attention is paid to practices and standards; medium resources where the spreadsheet model is made suitable for regular use; and high resources where the spreadsheet model has safeguards to the source code and inputs rendering it difficult to “break”.

We briefly discuss or sketch each of the nine types of analytical spreadsheet model in the taxonomy. We indicate the type of spreadsheets one might see for that type, and present formal inclusion criteria (with varying degrees of confidence) for some of them. We describe how these spreadsheets can be found in the literature. We share our perception of how easily they might be found in the field. We speculate on reasons (good and bad) that such spreadsheet models might be created in the rough-and-tumble of business, and discuss their relative riskiness and utility.

This taxonomy is best thought of as a coherent set of ideas that will be helpful to strengthening our understanding of existing spreadsheet development recommendations; the way that spreadsheets change over time; and an individual’s or organization’s decision-making around investing (or not) time and energy into spreadsheet models. There is much research that could be performed to test, explore, and deepen our understanding of this new use case-engineering resources taxonomy.

### 6.1 Testing and Refining the Use Case-Engineering Resources Taxonomy

We recognize that the domain of analytical spreadsheet models is very large, and it is highly ambitious to define a framework that can define the entire space. We are optimistic
that this taxonomy is useful. There is opportunity for further work on all aspects of the taxonomy. The engineering resource levels would benefit from better definition. The use case could be more deeply explored. All cells should have a set of criteria for membership, and their names might merit refinement.

The five cells that are not shaded in Figure 7 are less visible in the literature, and for reasons discussed in Section 5 are inherently problematic. These cells have potential for fruitful research.

For the IQASM cell (the center of Figure 7) we use the criteria from GMS. We suggest a set of criteria for Analytical Solution based on a reduction of the criteria for IQASM. Both of these sets of criteria should be tested against spreadsheets in the literature and in the field.

It would be desirable systematically to examine the field research on spreadsheets in light of this taxonomy, including Croll (2007) and Grossman, Mehrotra, and Özlük (2007). This taxonomy could usefully be examined through the lens provided by the unusually detailed FAST Standard (FAST 2019). It would also be helpful to examine this taxonomy in light of the largely-unpublished, rich, practical expertise of spreadsheet manufactories such as Operis Group, Ltd., F1F9, Modano (which seems to have absorbed and ceased support for SSRB/BPM), and undoubtedly many other organizations, and also in light of the guidelines and wisdom of practice available from e.g., ICAEW, F1F9’s delightful Resources page (F1F9 2023), etc.

It would be interesting to consider errors in analytical spreadsheet models in light of this taxonomy. Are there types of spreadsheets in this taxonomy that are particularly prone, or less prone, to error?

6.2 The Evolution and Transition of Spreadsheets Over Time

The development guidelines in the literature tend to be static. That is, they provide recommendations for a spreadsheet in a particular domain as a one-off development project. The framework has potential for to help us understand how spreadsheets change over time, and crafting recommendations for how they should evolve.

We suggest that there might be two ways that a spreadsheet might over time change its type in Figure 7. The first is purposefully, with due care, planning, and investment. We refer to this as transition from one type to another. The second is organically (or accidentally), where over time a spreadsheet shifts from one type to another type, without much in the way of forethought or purposeful investment. We refer to this change as evolution from one type to another.

There seems to be little literature on managing the rightward transition of a spreadsheet from Unique Analysis to Business Process-Embedded (which is a common transition), to Application (which we hypothesize is common). We hypothesize that in terms of Figure 7, a transition will tend to be a move up a row in the same column, or diagonally up and to the right, whereas an evolution would tend be a move to the right of the same row, reflecting a spreadsheet being distributed more broadly without additional engineering resources.

Guidelines for making smart engineering investments, with an eye to purposeful transition rather than organic/accidental evolution seem desirable. For example, sections 5.1 and 5.2 suggest that the transition would require engineering investment in modifiability, transferability, and readability. Indeed, Olavson and Fry (2008) discuss in detail the transition (to the right) from Analytical Solution to Industrial-Quality Analytical Spreadsheet Model. It would be valuable to generalize their insights beyond their domain of supply chain analytical models at Hewlett-Packard.
As another example, if an analyst has a Personal Productivity Spreadsheet (Unique Analysis Use Case & Low Engineering Resources) and realizes that it is becoming embedded in a routine business process (Use Case changes to Business Process-Enabled), what guidance could be provided for extending the spreadsheet up and to the right to become an Industrial-Quality Spreadsheet Model, hence avoiding the more risky outcome of becoming a Field Expedient?

7. ACKNOWLEDGMENT

We thank Professor Kourosh Dadgar for his helpful comments on a draft of this paper. We are grateful to two anonymous referees whose comments improved the paper. All errors of omission or commission are the responsibility of the authors.

8. REFERENCES


How Beaufort, Neumann and Gates met?
Subject integration with spreadsheeting

Maria Csernoch, Julia Csernoch

University of Debrecen, Faculty of Informatics
csernoch.maria@inf.unideb.hu, csernoch.julia@inf.unideb.hu

ABSTRACT

Computational thinking should be the fourth fundamental skill, along with reading, writing, and arithmetic (3R). To reach the level where computational thinking skills, especially digital problem solving have their own schemata, there is a long way to go. In the present paper, a novel approach is detailed to support subject integration and building digital schemata, on the well-known Beaufort scale. The conversion of a traditional, paper-based problem and a data retrieval process are presented within the frame of a Grade 8 action research study. It is found that both students’ content knowledge and their digital skills developed more efficiently than in traditional course book and decontextualized digital environments. Furthermore, the method presented here can be adapted to any paper-based problems whose solutions would be more effective in a digital environment and which offer various forms for building schemata both in the subject matter and informatics.

1. INTRODUCTION

1.1. Digitalization and digitization

According to Prensky [Prensky, 2001a] [Prensky, 2001b], digital students are overtaking the world. Among others, he also claims that digital natives are born with computers, consequently (1) they can develop on their own and they must do it on their own, since (2) education seems to be way behind them. However, research found proof that students do not have skills and abilities assigned to them by Prensky [Kirschner & De Bruyckere, 2017] [Csernoch et al., 2015], and developing their computational thinking skills [Wing, 2006] is a rather demanding and challenging task of education both from the side of students and teachers [Kirschner et al., 2006] [Lister, 2008]. The primary problem is how to engage students in solving real world problems, how they can find ways to meaningful digital solutions [Soloway, 1993], how they can control tools, instead of tools controlling them [Soloway, 1993] [Baranyi & Gilányi, 2013] [Baranyi et al., 2015] [Wolfram, 2020] [Wolfram, 2015]. On the other hand, it is also proven that education is way behind digitalization and digitization, and teachers are not prepared to fulfil their role in this novel era. All the aspects of TPCK (Technological Pedagogical Content Knowledge) [Angeli & Valanides, 2015] [Kadijevich et al., 2013] should be present in the teaching-learning process, but it is not so. In general, the problem we are faced with is complex, and it is high time to take some small steps. In our point of view, we must begin with finding the balance between classical and digital problem solving.

1.2. EDUCATION PRODUCTION SYSTEMS

In general, digital education is a push production system [Ohno, 1988] where tools, software interfaces, their novel features are in the focus, in the hope to build up huge knowledge inventory. However, the erroneous end-user products [Ben-Ari, 1999] [Ben-Ari & Yeshno, 2006] [Tufte, 2003] [Csernoch et al., 2022] [Csernoch et al., 2023], including spreadsheets [EuSpRIG, 2023] [Csernoch et al., 2015] [Csernoch et al., 2021] [Nagy et al., 2021], prove that this method cannot be either effective or efficient. Human mind is responsible for this
defect, since it does not work as a commodity inventory; knowledge, especially unconnected pieces of data cannot be stored in the long-term memory [Kirschner et al., 2006] [Sweller et al., 2011] [Low et al., 2011].

It is well known in the EuSpRIG community that time spent on fabricating documents, browsing, and clicking aimlessly use up an enormous amount of both human and machine resources. On the other hand, erroneous documents and their modification generate further waste that has serious financial consequences. We are convinced, in accordance with Ohno that the key is to give human intelligence to the machine (both hardware and software), and the transition from the single to the multi-skilled operators (end-users) [Ohno, 1988].

The present work is part of a larger campaign to introduce a digital education pull production system in accordance with the Toyota or Lean Production System (TPS) [Ohno, 1988] [Krafcik, 1988] [Womack & Jones, 2003] [Modig & Åhlström, 2018]. In this special education system, the values are to improve teachers’ and students’ computational thinking skills and problem solving abilities, furthermore to eliminate waste. The principles are just-in-time and autonomation which are supported by unorthodox methods and tools and activities. The just-in-time principle means that, instead of teaching software interfaces and features, problems are selected to solve, and only a limited set of tools is taught; those pieces which are necessary to solve the problem. This set includes data analysis, algorithms known from programming, and integration of software tools (the combination of various programs). Autonomation (automation with a human touch) is supported by the functional programming approach of Sprego [Csénoch, 2014] [Csénoch & Biró, 2015] [Csápo et al., 2019] [Csápo et al., 2020] and requires human intervention when the analysis and the debugging of the output reveal any discrepancy from standard.

The problem presented in the paper, selected from our depository, aims to demonstrate how this digital educational pull system works in classes.

1.3. Previous results

It is without question that there is high number of valuable papers and studies which details methods and results on developing students’ computational thinking skills with spreadsheet. However, it is beyond the scope of the present paper to cover thoroughly these sources, for at least two reasons. (1) The subject is developing and to cover the literature in a couple of sentences can miss the essence and serious works. (2) There is a restriction on the size of the paper, which does not allow both a classroom experience and a literature review.

In the present paper, we mention only those results which have strong connection with the detailed methods and tools. These statistical analyses proved that the classical tool-centred approaches (push production systems) do not develop students’ computational thinking skills, there is no knowledge built up in long-term memory [Csénoch et al., 2022] [Csénoch et al., 2023]. Furthermore, we also found proof that our pull system with the Sprego programming approach is much more effective than the tool-centred methods [Csénoch, 2014] [Csénoch & Biró, 2015] [Csápo et al., 2019] [Csápo et al., 2020]. The content awareness, which is our other concern, is proved effective and efficient in the works of the TPCK research [Angeli & Valanides, 2015] [Kadijevich et al., 2013]. Based on these results, we ventured into the subject integration field and found that while students work on a table, they get familiar with the content also. They learn local places and events, and make acquaintance with further subjects, like Beaufort scale (content selected for the present study), barely known in continental countries.

Considering connection between knowledge built up in long-term memory and the role of repetition, we found notable results in second language acquisition research. It is said in
language teaching that a word must be repeated over time which method is more effective than hearing a bunch of massed repetitions at one time. It is the same in computer sciences and informatics. While meeting the Beaufort scale in the printed course book only three times, in informatics classes, the word is mentioned so many times and the content is checked repeatedly that students learn both the name and the content. We have also found that to make students remember functions, they must be simple and be repeated in different contexts [Ghazi-Saidi & Ansaldo, 2017]. These reasons and findings explain the functionality of real world contents, repetition, and simplicity.

2. A CLASSROOM STUDY IN DIGITIZATION

2.1. The aims of the study

In the present paper, a classical paper-based problem is digitized, converted into a spreadsheet workbook within the framework of a classroom study [Stigler & Hiebert, 2009]. The research analysis reveals what skills can be developed by solving the course book tasks (Chapter 2.2), compared to the conversion process and the data management activities in spreadsheet with Sprego [Csernoch, 2014] [Csernoch & Biró, 2015] [Csapó et al., 2019] [Csapó et al., 2020].

Our ultimate goal is to prove that

- pull production system can be adapted in digital education and
- education production system can be as effective and efficient as it is in industry and service.

The target condition [Rother, 2010] [Liker, 2004] of this paper is to provide the essence of a classroom study which presents a method and its accompanying tools in a digital education pull production system. Further research is needed with novel partners (schools, companies, teachers, students, etc.) to statistically prove the effectiveness of the method.

In this education pull system, beyond developing teachers’ and students’ programming, data management, and spreadsheet knowledge, skills, and abilities, teaching and learning how to integrate sciences and school subjects and the normally (in push production systems) separately taught subjects of computer sciences have a great importance. With this method, both teachers and students learn the algorithms behind built-in, problem specific functions and spreadsheet features, and let go of software and/or software tools when problems require so. Two of the advantages are that (1) these fundamental algorithms and features can be used seamlessly (relying on schemata by activating fast thinking) [Kahneman, 2011] [Sweller et al., 2011] (2) and end-users have the background knowledge to select the most efficient tools to solve their problems (requiring slow thinking) [Kahneman, 2011] [Sweller et al., 2011] [Polya, 1945] [Smalley, 2018]. Furthermore, this system can reveal how classical, paper-based problems can be digitized which would serve a topic of data management, hardly covered yet.

However, we must keep in mind what Ohno said about TPS: “There is no real substance to that abstract mass we call ‘the public’. We discovered that industry has to accept orders from each customer and make products that differ according to individual requirements.” [Ohno, 1988]. In education, the public is the lot of students and end-users, and we must prepare dedicated classes and teaching-learning materials that suit their best interest. In these circumstances, all classes and students must be considered unique, but the shared results of classroom studies would help teachers in developing their teaching-learning materials.
2.2. Beaufort-scale

The task introduced in this paper arrives from a Grade 5 Sciences course book [Horváth et al., 2020], where a coloured Beaufort scale is presented (Figure 1), accompanied with three “classical” tasks.

- Use the table to determine the strength of the strongest wind you have ever encountered. Tell us what you experienced.
- What was the wind force described in the following short paragraph? Answer the question based on the Beaufort scale presented on page 110.
- Use the Beaufort scale to determine the approximate strength of the actual wind. If a suitable tool is available, measure the speed of the wind.

![Figure 1. The original Hungarian paper-based (left) and the Excel version (right) of the Beaufort scale.](image)

![Figure 2. An English version of the Beaufort scale [RMetS, n.d.].](image)
Several English Beaufort scale webpages are available on the internet (e.g., [Met Office, n.d.] [Edwards, n.d.] [US Department of Commerce NOAA, 2016] [RMetS, n.d.] [National Geographic, n.d.] [Encyclopædia Britannica, 2023] [Wikimedia Foundation, 2023]). In appearance and content, the closest to the course book version is [RMetS, n.d.] (Figure 2).

2.3. Concerning digitization

The primary aims of the present study are to provide the details of

- the conversion process from the original paper-based table to an Excel workbook,
- how the spreadsheet can support automated information retrieval, and
- how students can develop their skills and abilities in the TPCK process.

When the digitization of paper-based tasks or problems is in the focus, the main questions considering students’ development are:

- what competencies and skills can be developed,
- what can be stored in long-term memory,
- how subject integration can support deep learning,
- how fast and slow thinking can be activated in the data retrieval process.

Further questions are how teachers are prepared to be able to develop students’ content knowledge, their digital awareness, and their problem solving skills and abilities. In the following, a classroom study [Stigler & Hiebert, 2009] is detailed to present the digitization process of the selected content, the Beaufort scale in a Grade 8 group (13–14 years old students).

2.4. The description of the class

Students started to handle conversion processes in Grade 7 [Sebestyén et al., 2023] [Csernoch, 2019] (along with word processing), while spreadsheeting was launched at the beginning of Grade 8. Considering the content, at the beginning of the school year students confirmed that they never encountered Beaufort scale in advance to this project.

At the beginning of the spreadsheet studies, a Hungarian Beaufort scale webpage [Magyar Turizmus Média Kft., 2019] was converted to a spreadsheet workbook to become familiar with the content, to repeat webtable→datatable conversion [Sebestyén et al., 2023] [Csernoch, 2019], and to become familiar with fundamental spreadsheet concepts. Following this introductory section, the algorithm of linear search was introduced with lots of examples from real world situations (e.g., finding the row and the seat in a theatre; finding...
the platform, the wagon, and the seat at a train station; searching in a mobile list; unordered items) (Figure 3).

After linear search, examples of binary search on ordered lists are presented in real world situations (e.g., playing the Find the number! game, searching in a dictionary, and finding the proper-size screwdriver) (Figure 4).

Following the introduction of both algorithms, students can decide which algorithm should be used depending on the input data. This section took three classes in October.

The paper-based Beaufort scale was presented in the class in the second semester of the academic year. The students remembered the Beaufort scale and were delighted to meet a table which they were familiar with. The introduction of the content at the beginning of the year in a digital environment was enough to provide a firm background, based on which we were able to focus on the programming details. This section took six classes in January.

3. THE DATA MANIPULATION PROCESS

During the data manipulation process, the content of the webtable is converted into a 1st normal form datatable. The primary aims of the process are to make students

- become familiar with the content (again),
- recognize data fields, records, missing and redundant data,
- build the algorithm of the conversion process,
- find a suitable software environment or environments to carry out the planned process, and
- create an Excel-compatible format.

3.1. Analysis of the original table

The data management procedure starts with the analysis of the original table. In this step, the number of data fields and records should be identified, and the data types recognized. During this phase, students must keep in mind that the target format is a spreadsheet table. In this special table (originated from a printed course book), students should recognize that in the original first column three data are stored whose overall data type is string:

- data (content)
  - force,
  - the lower boundary of the speed,
  - the upper boundary of the speed,
- additional characters
  - parenthesis around the speeds,
  - n-dash between the lower and the upper boundary,
  - force and speeds are broken into two paragraphs.

Furthermore, in cells A2, A14, and C14 there is missing data (Figure 1, left panel).
In A2, there is only one speed.
In A14, there is only one speed with an additional < character.
C14 is empty.

The redundancy in the first column should also be recognized. Teachers must find ways to convince students that there is no need to store both the lower and the upper boundaries of the speed. Recognizing and handling redundancy is a complex problem, and it is always the teachers’ responsibility to introduce this concept, as early as possible [Hattie, 2012].

In general, the final Excel worksheet should contain 4 data fields, 1 row as header (field names) and 13 rows as data records. The data types in Columns A and B should be number, while in C and D string (Figure 1, right panel).

3.2. Conversion process: from PDF to DOCX

The conversion process starts with a file-save operation, where the page of the Beaufort scale is printed as a PDF file. This file can be opened in MS Word for further data handling (it could also be opened in Excel, however, for a beginner that task is too demanding). The left pane in Figure 5 shows the results of the conversion, and a compulsory first clean up phase which deletes data outside of the table.

The result is a tree-column table where some data are lost in the conversion process, but these minor errors can be corrected, along with the entry of the missing data (Figure 5, right).

3.3. Setting up data fields

After correcting and unifying the data in the first column (Figure 5, right panel), the separation of the three data – force, lower boundary and upper boundary of speed – and the deleting of the unnecessary characters – Enter, parentheses, and dash – must be performed. These steps are required before the proper data fields are set up.

The listed modifications can be completed with replacement, where
Enter and opening parenthesis are replaced with one Tab character (Figure 6, left), dash is replaced with one Tab character (Figure 6, middle), closing parenthesis is removed (replaced with nothing) (Figure 6, right).

With the Replace All command the number of replacements are presented in the information window, where we can check whether the number is correct or not. These numbers are 14, 13, and 14, respectively. In the first and the third replacements, the 13 records and the field name, while in the second, only the records contained the searched string.

The result of the data separation process is a table where two column separators are: tabulators and vertical lines (Figure 7). In the field-setup process, the next step is the unification of separators. Furthermore, we can decide whether we delete the upper boundary of speed in Word or we can leave it to Excel. In the presented solution, the first option is performed (Figure 8). The computer cooking (CoCoo for short) sequence is presented in CoCoo 1.
Figure 8. The data fields with a unified separator character, the tabulator.

the table with the mixed separators (Figure 7) is converted into text with Tabulator separator (Convert to Text) → the text is converted into a table of 5 columns (Convert Text to Table) → the column of the upper boundary of speed is deleted → the table is converted back to text (Convert to Text) (Figure 8).

CoCoo 1. The major steps of the unification of the field-separator characters.

3.4. Creating Excel workbook: from DOCX to XLSX

At this phase, the table is ready to be converted into an Excel workbook. One of the simplest ways to create an Excel workbook from a Word document is to use an interim file format, which is text file (Plain Text) with Unicode (UTF-8) encoding. The algorithm is presented in CoCoo 2.

CoCoo 2. The conversion process from Word document to Excel workbook.

3.5. Mapping

As a warm-up activity, a diagram was created, based on the data converted into an Excel workbook. The category labels are forces, while the values are the lower boundary of the speed (speed for short) (Figure 9).

Figure 9. Beaufort scale diagram.
4. COLOURING

The next step of the preparation process is the colouring of the Excel table (Figure 10, Columns A–D).

![Figure 10. The coloured Beaufort scale (Columns A–D) with the RGB codes (Columns E–G) and the picture of the original table (Column I). (Specifications are shortened in Column D.)](image)

To complete the colouring task, a picture was created from the original table of the course book (Figure 10, Column I). Based on the picture, the RGB codes of the colours (Figure 10, Columns E–G) were collected with a picker tool, which can be carried out either in a professional image manipulation program (usually we use GIMP, since it is available in our schools) or in a simple PowerPoint.

5. INFORMATION RETRIEVAL

5.1. Linear and binary searches

In the simplest retrieval tasks, we can provide the force or the speed as input (Figure 11, Cells F3 and F7, respectively) and write out the other three data. The difference between the two inputs is that the value of force can only be an integer from 0 to 12 (listed in the vector), while the speed can be any number greater or equal to 0. Consequently, the algorithm to find them in the corresponding vector is different (Force or Speed, respectively). In the case of speed, binary search must be used, while with force both linear and binary search can be applied. However, for students, linear search is more understandable, consequently it is recommended to use this simpler algorithm.

![Figure 11. Force (Cell F3) and speed (Cell F8) input values are entered by the user, and the other three values are calculated based on the Beaufort table.](image)

In both cases, the algorithm is provided in CoCoo 3.
finding the position of the input value in the corresponding vector (Formulas (1) and (3)) → writing out the value from another vector at the same position (Formulas (2) and (4))

CoCoo 3. The steps of the searching process.

|=MATCH(F3,A2:A14,0) (1)|
|=INDEX(B2:B14,MATCH(F3,A2:A14,0)) (2)|

Providing force as input, the linear search algorithm first finds the position of the given force in the corresponding vector (Force). After that, the speed is written out in the second step.

With the same match_type value (0), both the description and specification can be written out (Figure 11, Cells H3 and I3).

|=MATCH(F8,B2:B14) (3)|
|=INDEX(A2:A14,MATCH(F8,B2:B14)) (4)|

Providing speed as input, Formula (4) writes out the corresponding force (Figure 11, Cell G8). With the MATCH() function, using match_type=1 value (default value) a binary search is launched. With the same procedure, both the description and specification can be written out (Figure 11, Cells H8 and I8).

As the last step of these practices, complete sentences can be created with the combination of formulas and string constants (Figure 12).

5.2. Coloured sentences

All the output values can be coloured with the original colour of the corresponding force. To generate such outputs a conditional formatting is used with RGB codes gained from the original table (Figure 10).

Figure 12. Complete sentences in Cells F4 and F9 with the given input values in Cells F3 and F8.

Figure 13. Complete sentences with the corresponding colour of force.
5.3. Drop-down lists

In the present paper details of drop-down lists created to the description vector is presented. However, in a similar way drop-down lists can be created for the specifications and even for the forces.

To select descriptions, drop-down lists are more convenient than typing these long strings. Two types of objects can be inserted into the sheets:

- Form Controls (Figure 14, Cell G3),
- ActiveX Controls (Figure 14, Cell G10).

In both cases, two parameters of the object must be set up:

- Input range (ListFillRange), the values (items) to appear in the list (Figure 10, Range C2:C14),
- Linked cell (LinkedCell), the cell displaying the output (Figure 14, Cells F3 and F10).

The difference between the two types of object is the output:

- Form Control returns the position of the item in the vector (Figure 14, Cell F3) (the index of the selected item),
- ActiveX Control returns the selected item (Figure 14, Cell F10).

![Figure 14. Drop-down lists for selecting specification.](image)

Based on the selected description, the other data of the records can be presented in the table. To write out the specification for the item selected from a Form Control a simple INDEX() function serves our purposes (Figure 14, Cell G5) (Equation (5)).

\[ =\text{INDEX(D2:D14,F3)} \]  
(5)

To create a complete sentence with the selected description and the corresponding speed, two INDEX() functions (Formulas (5) and (6)) can be called with the additional string constants (Figure 14, Cell G6).

\[ =\text{INDEX(B2:B14,F3)} \]  
(6)

In the case of ActiveX Control, to write out the specification (Figure 14, Cell G12) the 2-level function of INDEX(MATCH()) can be called (Formula (7)).

\[ =\text{INDEX(D2:D14,MATCH(F10,C2:C14,0))} \]  
(7)
To create a complete sentence with the selected description and another data from the record (Figure 14, Cell G13) a reference to the output cell (LinkedCell) (Formula (8)) and the 2-level INDEX(MATCH()) can be called (Formula (7)), with the additional string constants.

```
=F10
```

Capitalizing the first letter of the sentence requires some text handling, whose algorithm is presented in CoCoo 4.

```
separating the first letter of the description → changing the lower-case letter to capital → separating all the characters except the first one of the description from the right side → concatenating the two strings (Formula (9)) (Figure 14, Cell G13)
```

CoCoo 4. The algorithm of capitalizing the first letter of the sentence.

```
=UPPER(LEFT(F10))&RIGHT(F10,LEN(F10)-1)
```

The outputs based on a selection from a drop-down list can also be coloured, similar to sentences presented in Figure 13.

6. SUMMARY

6.1. Manual search in the table

In the original course book tasks (Chapter 2.2), students were required to carry out three manual searches in the presented table to provide two times the speed and once the force of the wind experienced or described in the book. The tasks would take five minutes. Beyond completing these manual searches in the Beaufort scale, students never encounter the table again.

6.2. Digitization of the table

In the digitization process of the table, students carry out several different activities, and learn how to

- set up a data table (1st normal form), fields, records, column separators,
- analyse and correct data,
- enter the escape sequences of non-printable characters in Word,
- convert text to table and table to text in Word,
- handle different data types (in this task: webpage, pdf, Word document, text file, Excel workbook, image),
- copy and name worksheet, freeze panes,
- insert and resize image on a worksheet,
- read RGB codes of an image,
- create and format diagram,
- enter input data with methods which are more secure and faster than typing (dropdown lists),
- create multi-level functions,
- handle n-ary functions,
- use the algorithm of linear and binary searches,
- create flexible natural language sentences with variables and formulas,
- set up conditional formatting,
- carry out debugging.
The digitization of the course book table was introduced with the conversion of a Beaufort webpage. The two processes took nine classes together. During these classes, students had the opportunity to get familiar both with the name and the content of the table and learn and practice the basis of functional programming in spreadsheet.

6.3. Improvement of spreadsheet quality

The essence of a digital education pull production system is that only those fundamental knowledge pieces and tools are introduced and taught which are needed to solve the presented problems, matching the characteristics of the students (end-users). With this method in the first learning period, slow thinking is activated and schemata are built up (standardization) [Sweller et al., 2011] [Kahneman, 2011] [Polya, 1945]. Later, fast thinking can activate schemata to fast and reliable problem solving. Considering these roles of fast and slow thinking in spreadsheet education, we provide some examples connected to already existing features.

E.g., pivot table is an excellent tool. However, it is a horror for those who do not understand the algorithms behind it. To avoid this discrepancy, in Sprego first, we teach and thoroughly practice conditional calculations along with conditional formatting then we can try pivot tables. On the same background with conditional operations, we can eliminate the specified conditional and database functions with all their restrictions.

E.g., the table in the table feature is one of the newest and useless features in education. If end-users learn how to create 1st normal form table, there is no need for the Format as table command. Furthermore, knowledge connected to normal forms can be transferred to database management and programming.

E.g., if students learn fundamental algorithms in spreadsheet environment, later they can transfer this knowledge to ‘serious’ programming.

It is said at Toyota that “Take Good Care of Old Equipment. One source of eliminating waste is not to buy new computers (digital devices) just because they appeared on the market. … A machine’s value is not determined by its years of service or its age. It is determined by the earning power it still remains.” [Ohno, 1988]. This saying is true with computer sciences and informatics; just because a new hardware, software, or a new feature in a software appears, there is no need to rush head over heels to get it. Teaching fundamental algorithms and general-purpose functions allow us to create program- and version-independent spreadsheets that can be applied in any interface, while it is not true with fancy new features.

7. DISCUSSION

Beyond the traditional sense of subject integration, novel concepts, including TPCK [Angeli & Valanides, 2015] [Kadijevich et al., 2013] and digital problem solving, slowly though, but are advancing and reaching education. Within this framework, another subject integration can be introduced, namely the connection and knowledge-transfer between the various subjects of informatics (computer sciences). The main difference between the two types of subject integration is that this latter often raises debates and is seldom accepted in spite of its unquestionable values.

In the present paper, an example of digital subject integration is detailed. The subject is the Beaufort scale of winds – adapted from a traditional Grade 5 Sciences course book. The original paper-based table is converted into an Excel workbook, and in this GUI simplified functional programming is performed (exaptation, defined in [Hatamleh & Tilesch, 2020]). The paper details the data analysis and the conversion process, and provides examples of data retrieval methods in spreadsheets, along with the algorithms concerned.
It is found that the introduction of the selected table in a digital environment helps students to deepen their content knowledge in sciences and offer a real-world problem for programming, data management, and information retrieval. During the data analysis and handling process, especially in the discussion section, students must be aware of both the concept and the content of the table. Furthermore, the conversion process requires skills that strengthen the connection between different data management approaches and tools. The combination of these skills deepens students’ knowledge in both sciences and informatics, which is one of the aims of education.

In general, the presented method is part of a huge project which ultimate goal is to find ways to set up a digital education pull production system. The TPS pull production system [Ohno, 1988] [Krafcik, 1988] [Womack & Jones, 2003] [Modig & Åhlström, 2018] is proved efficient in industry and service, so we are convinced that the low-growth period of digital education can benefit from such changes.

REFERENCES


Hatamleh, O., & Tilesch, G. (2020). BetweenBrains; Taking back our AI future. GTPublishDrive.


**SOURCES**


---

1 https://pixabay.com/images/id-1727890/ 12.30pm 23/05/27
2 https://pixabay.com/images/id-911760/ 12.30pm 23/05/27
3 https://pixabay.com/images/id-7304257/ 12.30pm 23/05/27
4 https://pixabay.com/images/id-1044718/ 12.30pm 23/05/27
5 https://pixabay.com/images/id-2001129/ 12.30pm 23/05/27
6 https://cdn.pixabay.com/photo/2014/11/09/15/35/the-screwdriver-523783_1280.jpg 12.30pm 23/05/27
7 https://pixabay.com/images/id-3509492/ 12.30pm 23/05/27
8 https://pixabay.com/images/id-1798/ 12.30pm 23/05/27
9 https://pixabay.com/images/id-2134715/ 12.30pm 23/05/27
Reducing Errors in Excel Models with Component-Based Software Engineering

Craig Hatmaker
Craig_Hatmaker@Yahoo.Com

ABSTRACT

Model errors are pervasive and can be catastrophic. We can reduce model errors and time to market by applying Component-Based Software Engineering (CBSE) concepts to Excel models. CBSE assembles solutions from pre-built, pre-tested components rather than written from formulas. This is made possible by the introduction of LAMBDA. LAMBDA is an Excel function that creates functions from Excel’s formulas. CBSE-compliant LAMBDA functions can be reused in any project just like any Excel function. They also look exactly like Excel’s native functions such as SUM(). This makes it possible for even junior modelers to leverage CBSE-compliant LAMBDA to develop models quicker with fewer errors.

1 INTRODUCTION

This paper opens by discussing Excel as a programming language that makes applying CBSE concepts to Excel possible. Section 3 discusses CBSE’s value to both modelers and end users. Section 4 explains what qualifies a component as CBSE compliant. Section 5 discusses how component creators can make their components discoverable and how component consumers can include them in their projects. Section 6 walks us through assembling a real-world project. Section 7 summarizes this paper and is followed by references. An appendix provides answers to questions I have encountered and questions I expect from the modeling community.

2 EXCEL AS A PROGRAMMING LANGUAGE

This section discusses Excel as a programming language. Some of this paper’s references use computer science terms. These terms may be unfamiliar to many Excel professionals so the table below may be helpful.

<table>
<thead>
<tr>
<th>CBSE Term</th>
<th>Excel Equivalent</th>
</tr>
</thead>
<tbody>
<tr>
<td>Component</td>
<td>An Excel workbook, worksheet, chart, or function whether native, user-defined or from add-ins.</td>
</tr>
<tr>
<td>Component Consumer</td>
<td>Someone who uses components</td>
</tr>
<tr>
<td>Component Creator</td>
<td>Someone who develops CBSE compliant LAMBDAs.</td>
</tr>
<tr>
<td>Component Interface</td>
<td>A function’s parameters (aka function arguments)</td>
</tr>
<tr>
<td>Defect Density</td>
<td>Errors per formula cell</td>
</tr>
<tr>
<td>Developer, Programmer</td>
<td>Modeler</td>
</tr>
<tr>
<td>Line of Code</td>
<td>A formula in a cell</td>
</tr>
<tr>
<td>Programming Language</td>
<td>Excels functions, operators, references, and formula syntax rules</td>
</tr>
<tr>
<td>System, Application</td>
<td>Completed model</td>
</tr>
</tbody>
</table>
Excel is the world’s most widely used programming language [Microsoft 2021]. Though long accepted as a programming language, Excel lacked features that would qualify it as a Turing-complete programming language until the introduction of LAMBDA. As a Turing-complete programming language, Excel users can, in principle, write any computation. [Microsoft 2023].

LAMBDA was announced in December 2020 and made generally available in February 2022. It is an Excel function with which users can define new functions from Excel formulas. Before LAMBDA, creating user-defined functions (UDFs) required knowledge of programming languages like VBA. Because LAMBDA’s programming language is normal Excel formulas, anyone who can write Excel formulas can also create UDFs. The LAMBDA programming language includes all Excel functions but also features unique to LAMBDA as an Excel function that greatly expands Excel’s capabilities.

**Recursion**

Recursion is a feature LAMBDA shares with most programming languages but not with any other Excel function. Recursion is the ability for a function to call itself and adds the ability to process hierarchical data structures to Excel. A simple hierarchical data structure example is the Fibonacci sequence. The Fibonacci sequence starts with 0 and 1. The next number in the sequence is the sum of its two predecessors; thus, the third number is 1 (0 + 1), the fourth number is 2 (1 + 1) and the third number is 3 (1 + 2).

To calculate the \( n \)th number, we can write an elegant recursive formula like the one in Figure 1.

![Figure 1 Recursive LAMBDA displayed in Excel's Name Manager](image)

To use it in a cell to find the 9th number in the Fibonacci sequence we would write =Fibonacci(9). The function is intuitive, and the formula is compact.

Recursion can be simulated by enabling circular equations via Excel’s menu path File > Options > Formulas > Enable iterative calculation and setting the maximum iterations. Unfortunately, this enables all circular equations, even ones we did not intend to be circular. Recursive LAMBDA s do not need this option checked and can limit recursion to the confines of a single LAMBDA. LAMBDA recursion has no impact on other formulas, and it does not permit LAMBDA s with external circular references to iterate.
LAMBDA’s Helper Functions

LAMBDA has several helper functions unique to it including BYROW, BYCOL, MAKEARRAY, MAP, REDUCE, and SCAN. These make dynamic arrays work when they would not otherwise. Dynamic Arrays by themselves can reduce error potential by addressing inconsistent formulas which is a common spreadsheet error [O’Beirne 2010] [O’Beirne 2014] [Mkamanga K 2021]. With dynamic arrays, one formula entered into one cell can spill into tens of thousands of other cells which simply cannot be inconsistent. These functions with LAMBDA greatly expand how we can work with dynamic arrays to reduce errors and speed delivery.

Another function that can be used without LAMBDA, but which helps enormously is LET(). LET can mimic a feature shared by all programming languages like VBA known as local variables. Local variables can only be accessed within the function. This means they cannot interact with other functions even if those other functions have a variable of the same name. LET’s version of local variables is locally defined named formulas. Locally defined named formulas are very similar to named formulas defined in Excel’s Name Manager but as locally defined named formulas, they never appear in name manager. Another advantage of locally defined formulas is they can segment complex formulas into smaller, easier-to-understand, and easier-to-test sections with their intent conveyed by their name.

2.1 Section Summary

Excel is a programming language that everyone writing formulas uses. LAMBDA extends Excel’s programming language to make it Turing-complete like other programming languages such as VBA. As a Turing-complete programming language, we can leverage the vast knowledge base for Turing-complete languages compiled over many decades such as CBSE.

3 COMPONENT-BASED SOFTWARE ENGINEERING VALUE PROPOSITION

Component-based software engineering (CBSE), also known as component-based development (CBD), emerged in the late ’90s as an approach to software systems development based on reusing software components [Sommerville-2016]. Microsoft Excel is an example of a component-based application [Brown, 2000]. CBSE’s value promise includes improved solution complexity management, increased productivity, reduced time to market, improved consistency, improved quality, and reduced maintenance costs, among other things. [Ivica 2001] [Brown [2000]. While CBSE provides many benefits, this paper concentrates on just two: reducing errors, and time to market.

3.1 Errors and Productivity

Model errors are pervasive [Panko, 2005] and can be catastrophic [O’Beirne 2023]. CBSE addresses errors by leveraging pre-tested reusable components. The theory is components designed for reuse will be tested more thoroughly before implementation [Mohaghegh 2004]. Additionally, errors will be exposed as a component is used in varied situations. Once exposed the function will be remediated and then placed back into production. This process repeats until no more errors surface. The monitor, fix, and repeat cycle is not unique to the software industry.
It is also part of many manufacturing quality assurance best practices including CIP (Continual Improvement Process) [Wikipedia 2023-1], TQM (Total Quality Management) [Wikipedia 2023-2], and Six Sigma [Wikipedia 2023-2].

In the software industry, this theory has been tested by several researchers. Researchers from the MITRE corporation measured the impact of code reuse and defect density. They determined high reuse reduced errors by as much as 50% [Agresti 1992]. Researchers in Norway conducted similar research and also concluded that reused components resulted in nearly 50% fewer defects [Mohaghegh 2004]. A study in 1995 found an even stronger reduction in errors [Melo 1995].

I have found this to be true in my own experience as a developer, I have dramatically reduced errors and my time to deliver through code reuse. Solutions that I estimate would have taken me many months to code, took me weeks by assembling components from purchased libraries. Researchers have measured the correlation between code reuse and productivity and found timesaving not just from reusing components instead of coding them, but also in a reduction in testing and rework due to the higher quality of pre-tested components. [Melo 1995].

3.2 Additional benefits

LAMBDA components can codify business logic. Business logic is the expression of business rules in code. For example, if a company has a rule on which DCF (Discount Cash Flow) method they use and which variation of WACC (weighted average cost of capital) they use to calculate DCF, that formula can be placed into a reusable component. Then, the company can make that rule accessible to its modelers and allow only that component to be used in their projects.

3.3 Section Summary

CBSE has been proven to reduce errors and speed delivery. It can also be used to enforce business rules in models.

4 WHAT IS A CBSE COMPONENT?

The previous section explains why CBSE is important to modeling. This section explains what CBSE is within the Excel context.

There are several definitions for a CBSE component that I will interpret and consolidate.

A CBSE component is:
- designed for reuse without modification
- within the Excel application
- providing non-trivial functionality
- which is accessible through its function parameters/arguments.
- It is deployed as a self-contained independent object
- Ideally, it should be well-documented,
- provide user feedback and be rigorously tested
Excel’s native functions provide over 500 examples. Each function is designed to be reused without modification many times within the same Excel workbook and other Excel workbooks. Each function provides non-trivial functionality that we cannot modify, and that we can only interact with through their parameters (arguments). They are all well documented through online help and rigorously tested.

4.1 Creating CBSE-Compliant LAMBDA

In the CBSE context, there are two classes of users, component creators and component consumers. A component creator is someone who develops CBSE-compliant LAMBDA. Component creators are typically highly skilled though that is not always a requirement. Component consumers are those who use components without having to know how the components work. This section is for component creators.

One CBSE requirement is the function must be non-trivial. While it is possible to write trivial LAMBDA, the cost to create reusable components meeting all requirements can only be justified through reuse [Frakes, 1996]. So let us look at the extra work required to make LAMBDA function reusable.

The LAMBDA function has three sections: parameter names, the calculation, and parameter values.

How component creators use these sections determines if a LAMBDA can be reused everywhere. To be reused everywhere LAMBDA must be self-contained [Szyperski 2002]. To be self-contained, the LAMBDA calculation section must not reference cells directly as that would make them dependent on the host worksheet. Table 4 provides examples of the CBSE-compliant way to incorporate cell references (row 2) and the non-compliant way (row 3). The compliant way passes cells as parameters so we can pass these values without modifying the LAMBDA’s calculation section. The non-compliant LAMBDA requires the values in cells A2 and B2. The non-compliant formula will break if we move the values to any other cells.

<table>
<thead>
<tr>
<th>Parameter Names</th>
<th>Calculations</th>
<th>Parameter Values</th>
</tr>
</thead>
<tbody>
<tr>
<td>Parm1, Parm2</td>
<td>=LAMBDA([ParmName1, ParmName2, …], Calculation)((ParmValue1, ParmValue2,…))</td>
<td></td>
</tr>
</tbody>
</table>

Table 1 CBSE compliant (green) and non-compliant (red) LAMBDA

Table 1 CBSE compliant (green) and non-compliant (red) LAMBDA

= LAMBDA([ParmName1, ParmName2, …], Calculation)((ParmValue1, ParmValue2,…))
This rule also applies to LAMBDA’s helper functions: BYROW(), BYCOL(), MAKEARRAY(), MAP(), REDUCE(), and SCAN(). Table 5 provides examples of the CBSE-compliant way to incorporate cell references (row 6) and the non-compliant way (row 2). The compliant way passes cells as parameters which requires wrapping these formulas in an outer LAMBDA. This adds a little more work.

<table>
<thead>
<tr>
<th>A</th>
<th>B</th>
<th>C</th>
<th>D</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Parm</td>
<td>Result</td>
<td>Formula Text</td>
</tr>
<tr>
<td>2</td>
<td>1</td>
<td>2,1,2</td>
<td>=BYROW({A2:B4, LAMBDA(Row, TEXTJOIN(,, TRUE, Row))})</td>
</tr>
<tr>
<td>3</td>
<td>3</td>
<td>4,3,4</td>
<td></td>
</tr>
<tr>
<td>4</td>
<td>5</td>
<td>6,5,6</td>
<td></td>
</tr>
<tr>
<td>5</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>6</td>
<td>1</td>
<td>2,1,2</td>
<td>=LAMBDA(Array, BYROW({Array, LAMBDA(Row, TEXTJOIN(,, TRUE, Row))}){A6:B8})</td>
</tr>
<tr>
<td>7</td>
<td>3</td>
<td>4,3,4</td>
<td></td>
</tr>
</tbody>
</table>

Table 2 CBSE compliant (green) and non-compliant (red) LAMBDAs with helper functions

LET() is not unique to LAMBDA but to create CBSE-compliant LAMBDAs that use LET, LET must not contain references to cells or named formulas. The diagram below shows the LET function in row 2 directly accessing dynamic arrays A2# and B2#. This is not CBSE-compliant. To be CBSE compliant we can wrap the LET() in a LAMBDA and receive the dynamic arrays via the LAMBDA’s parameters.

<table>
<thead>
<tr>
<th>1</th>
<th>Parm</th>
<th>Result</th>
<th>Formula Text</th>
</tr>
</thead>
<tbody>
<tr>
<td>2</td>
<td>1</td>
<td>2,3</td>
<td>=LET({</td>
</tr>
<tr>
<td>3</td>
<td>2</td>
<td>3</td>
<td>Array, HSTACK(A2#, B2#),</td>
</tr>
<tr>
<td>4</td>
<td>3</td>
<td>4</td>
<td>BYROW({Array, LAMBDA(Row, SUM(Row))})</td>
</tr>
<tr>
<td>5</td>
<td>5</td>
<td>5</td>
<td>9</td>
</tr>
<tr>
<td>6</td>
<td>6</td>
<td>6</td>
<td>11</td>
</tr>
<tr>
<td>7</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>8</td>
<td>1</td>
<td>2,3</td>
<td>=LAMBDA(Column1, Column2,</td>
</tr>
<tr>
<td>9</td>
<td>2</td>
<td>3</td>
<td>LET(Array, HSTACK(Column1, Column2),</td>
</tr>
<tr>
<td>10</td>
<td>3</td>
<td>4</td>
<td>BYROW({Array, LAMBDA(Row, SUM(Row))})</td>
</tr>
<tr>
<td>11</td>
<td>4</td>
<td>5</td>
<td>9</td>
</tr>
<tr>
<td>12</td>
<td>5</td>
<td>6</td>
<td>11 ){A8#, B8#}</td>
</tr>
</tbody>
</table>

Table 3 CBSE compliant (green) and non-compliant (red) LAMBDAs with LET
### 4.2 Documenting Components

CBSE-compliant LAMBDAs require documentation that forms the component specification. The documentation must include a description of the services it provides, a list of allowable inputs with their constraints, a description of outputs, and any conditions for which the component will and will not function. [Brown 2000] [Szyperski 2002]. Below is an example:

```latex
// Function Name: SumColumnsA
// Description: Totals an array's columns.
// Revision: Original Development
// Date: Mar 06 2023
// Developer: Craig Hatmaker

SumColumnsA = LAMBDA([Array],
  LET(
    Help,
      TRIM(TEXTSPLIT(
        "DESCRIPTION: Creates totals for each column in an array."
        "VERSION: Apr 10 2023" &
        "PARAMETERS:: Array (Required) A two dimensional array of values."
        "EXAMPLES: 15,25,35,45 =SumColumnsA([1,2,3,4;4,3,2,1;10,20,30,40])",
        "-", "-" ),

    // Check inputs
    Array, IF(OR(ISOMITTED(Array), Array=""), #Value!, Array),
    // Procedure
    Result, BYCOL(Array, LAMBDA(Column, SUM(Column))),
    // Handle Error
    Error, MAX(IsError(Result)+1),
    // Return Result
    Choose(Error, Result, Help)
  ));
```

Figure 2: Example CBSE compliant documentation for LAMBDAs
of a LAMBDA source with documentation meeting these requirements as displayed in the Advanced Formula Environment.

4.3 Section Summary

Converting good LAMBDA to CBSE-compliant LAMBDA requires a little extra work which centers on moving all cell and named formula references from the LAMBDA’s calculations section, helper formulas, and LET functions to the LAMBDA’s parameters section. Good documentation is also required. Adding features like feedback is ideal.

5 USING CBSE-COMPLIANT LAMBDAS

Component creators need to store their components in a repository they can write to, and component consumers can read from. For components codifying business logic, the repository can be a network directory that only members of the company can access. If the component creator wishes to freely distribute their LAMBDAs, a free repository is a public GitHub Gist (https://Gist.GitHub.com) which Microsoft’s free Advanced Formula Environment (AFE) add-in for Excel integrates with to import LAMBDAs directly into Excel. AFE also integrates with private GitHub Gist which component creators can use for limiting access.

Component consumers need the means to find components. At the time of this writing, there are very few CBSE-compliant LAMBDAs available, and none can be found by Google search. To find them, component consumers can go to Gist.GitHub.com and use its search facility with the search string “CBSE compliant LAMBDA”. Gists designated as “public” can be discovered this way. Gists designated as “private” can only be accessed by URL. An option to find private Gists exists at Eloquens.com which has a market for LAMBDAs.

Once a desired component is found and its URL obtained, component consumers can use Microsoft’s free Excel add-in, Advanced Formula Environment (AFE) [William 2023]. AFE is designed to import LAMBDAS Gists referred to as modules by AFE. AFE imports modules, directly into Excel. To avoid the potential of imported LAMBDAs having the same name as other items in Name Manager, AFE asks consumers to provide a short ID for the module which AFE applies as a prefix to each LAMBDA from that Gist. AFE stores the imported LAMBDAs in the workbook’s XML and makes them available within that workbook like any other Excel function.

Gists can be updated either to correct a problem or improve functionality. To preserve backward compatibility, previous version interfaces (parameters) must not change but new parameters can be added to the end of the interface definition as optional. When changed, GitHub logs the changes in detail and increments the version number. Component Consumers can check their version against what is available in GitHub to determine if they have the most current version, and if not, what the differences are.
5.1 Section Summary

Component creators can store components freely in GitHub Gists. Component consumers can find public components using the GitHub Gists search facility. Once found, component consumers can use AFE’s import feature to load modules directly into Excel where they can be used like any other Excel function. Example

The following example is for building and managing a departmental budget. Notable features include:

1. It can handle any number of accounts, budget items, actual expenditures, and periods without modeler intervention.
2. No refresh is required (except to download actuals if using Power Query).
3. Calculations can be assembled in 6 minutes.

It uses two LAMBDA libraries:

1. Excel CBSE Compliant LAMBDA Dates (module prefix used BXD):
   https://gist.github.com/CHatmaker/3e1708888ec2bd1cde2cc9d002dc459b
2. Excel CBSE Compliant LAMBDAs for Reporting with Arrays (module prefix used BXR):
   https://gist.github.com/CHatmaker/cc0e8975d30b40734641a06dbae02143

The template with worksheets and modules but no calculations can be obtained here:
https://www.dropbox.com/s/8wtwexl8sarkdb2/CBSE%20Budget%20Template%20Blank%20with%20LAMBDAs.xlsx?dl=1

The completed example can be downloaded from here:
https://www.dropbox.com/s/j82p9v6vbmo5q8z/CBSE%20Budget%20Template%20Completed.xlsx?dl=0

A video showing the end-user experience of this model is available here:
https://youtu.be/uEXd8h1i3LM

A video showing assembling this model is available here:
https://youtu.be/ERzF2uw4cX4

The add-in I used in to simplify naming dynamic arrays is available here:
https://www.eloquens.com/tool/ELyYf0m4/finance/excel-tips/bxl-dan-free-create-dynamic-array-names-and-more
5.2 Worksheet Inputs

We start with the user inputs worksheet which has sample data already loaded. On the left are the model’s properties which include the Start date, End date (blank defaults to one year), and reporting period Interval. These are named cells. On the right are budgeted items. Each item has an Account, amount (Ext.Amt.), Schedule code, and start date (First Date). Some items have an end date (Last Date). The schedule code designates whether a budget item repeats at regular intervals which include daily, weekly, biweekly, monthly, quarterly, semiannually, annually, or not at all.
5.3 Worksheet ItemSchedule

Our project’s calculations start by scheduling all budget item amounts on worksheet ItemSchedule.

For this, we will use components related to scheduling and dates, and components related to reporting that perform grouping and summing functions. The components we need are already loaded into the template. I named the reporting components’ module BXR. I named the date components’ module BXD. We can see these components using the AFE. See the Appendix for answers on how to load and access the AFE.

These two modules contain an About component. If we enter =BXD.About into a cell we get information about the dates module and a component listing. See below.
The components we need from module BXD (dates) module are: BXD.CreateStartDates() which creates a timeline, and BXD.IsOccurrenceDate() which determines if a date in the timeline is one upon which a budget item occurs. Below shows these components in their locations.

Before we turn our attention to the components, let us bring in the labels from our budget items table. Enter this formula in cell A7: =tblBI[[ID];[Expense Description]]. At this point we should see an array of the ID through Expense Description columns from tblBI (Budget Items table on the Inputs worksheet).
These labels are not part of our calculations. They are there just to help us relate calculated amounts to their associated budget item entries.

Let us now turn our attention to components. Because these are CBSE components, they include user feedback. One form of feedback is inline help. If we go to a blank worksheet and type into any cell: =BXD.CreateStartDates\(\lambda()\), the inline help spills into empty cells.

<table>
<thead>
<tr>
<th>DESCRIPTION:</th>
<th>Creates a horizontal timeline or period start dates.</th>
</tr>
</thead>
<tbody>
<tr>
<td>PARAMETERS:</td>
<td></td>
</tr>
<tr>
<td>StartDate</td>
<td>(Required) First date in timeline.</td>
</tr>
<tr>
<td>EndDate</td>
<td>(Optional) Last date in timeline. Defaults to 1 year after start.</td>
</tr>
<tr>
<td>Interval</td>
<td>(Optional) Difference in Days, Weeks, Months (default), Quarters, or Years</td>
</tr>
<tr>
<td>REQUISITES:</td>
<td>Requires Periods(\lambda()) from this LAMBDA’s source module</td>
</tr>
<tr>
<td>EXAMPLES:</td>
<td></td>
</tr>
<tr>
<td>Result</td>
<td>Formula</td>
</tr>
<tr>
<td>1 yr in Months</td>
<td>=CreateStartDates(“1/1/2023”)</td>
</tr>
<tr>
<td>1 yr in Days</td>
<td>=CreateStartDates(“1/1/2023”, “D”)</td>
</tr>
<tr>
<td>2 yrs in Qtrs</td>
<td>=CreateStartDates(“1/1/2023”,“12/31/2024”, “Q”)</td>
</tr>
</tbody>
</table>

**NOTE!** We do not need to enter the lambda symbol (\(\lambda\)), nor the module prefix. I generally just start typing and by the time I reach CreateS Excel’s intellisense autocompletes the formula for me and I can just hit ENTER.

Let us create a timeline. On our ItemSchedule worksheet enter this formula in cell F6: =BXD.CreateStartDates\(\lambda(\text{Start}, \text{,”D”})\).
• *Start* is the named range on the *Inputs* worksheet which the user can use to designate when their budget year starts. For rolling budgets, this can be changed monthly and the model will automatically adjust.

• *End* date is omitted and will default to one year.

• “D” tells our component to set the timeline interval to daily.

At this point, we see a daily timeline extending out one year.

Let us name the timeline by selecting cell F6 and then use Excel ribbon tab Formulas > Define Name. Enter *Dates* in the Name: box and add a hashtag (pound symbol) in the Refers to: box after =ItemSchedule!$F$6. The hashtag makes the name refer to the array instead of just the cell. Click OK.

To schedule our budget amounts, enter this formula in cell F7:

=BXD.IsOccurrenceDate(Dates, tblBI[First Date], tblBI[Last Date], tblBI[Schedule]).

• *Dates* is our timeline.

• *tblBI[First Date]* is the *First Date* column from our budget items table on our *Inputs* worksheet.

• *tblBI[Last Date]* is the *Last Date* column.

• *tblBI[Schedule]* is the *Schedule* column.
At this point, we see an array of TRUE and FALSE where TRUE is when a budget item’s expense occurs.

We can replace the TRUE and FALSE values with budget amounts by multiplying the formula by tblBI[Ext.Amt.]. Change the formula to this: 

\[ =BxD.IsOccurrenceDate(Dates, tblBI[First Date], tblBI[Last Date], tblBI[Schedule]) \times tblBI[Ext.Amt.] \]

At this point, we see an array of budget amounts.

Let us name this array ItemSchedule by selecting cell F7 and then use Excel ribbon tab Formulas > Define Name. Enter ItemSchedule in the Name: box and add a hashtag (pound symbol) in the Refers to: box after =ItemSchedule!$F$7. Click OK.
We total each row by entering the SumRowsλ component from module BXR. Enter this formula in cell E7: =BXR.SumRowsλ(ItemSchedule). At this point we see an array of totals for each budget item.

Lastly, we will total the entire array which we will use to compare against other sections of this model to confirm each budget section adds up to the same amount. In cell A5 enter this normal Excel formula: =SUM(ItemSchedule).
5.4 Worksheet ItemsByPrd

The next sections are assembled in a similar fashion. This section’s purpose is to summarize budget items by reporting period (months in this case) instead of by day. It uses four components. Three are new: BXR.SumColumns(); BXD.PeriodLabel(); and BXR.SumGroups().

Below are the results after entering these formulas. I named the timelines: **PeriodStarts** and **PeriodEnds**. I named the main array: **ItemSummary**.
5.5 Worksheet BudgetByAcctPrd

This section’s purpose is to summarize budget items by account instead of by item. It uses five components. Three are new: BXR.SumColumnsλ; BXD.PeriodLabelλ, and BXR.SumGroupsλ.

Below are the results after entering these formulas. I named the timelines: Periods. I named the main array: BudgetSummary.
5.6 Worksheet ActualsByAcctPrd

This section’s purpose is to transform and summarize actual expenditures imported from our financial system. It uses three components. One is new: BXR.SumGroupsAndPeriodsλ. This component pivots the tabular data, then summarizes by account and period.

Below are the results after entering these formulas. I named the main array: ActualsSummary.
5.7 Worksheet BalancesByAcctPrd

This section’s purpose is to calculate balances by subtracting actuals from budgeted. It uses two components. None are new.

Below are the results after entering these formulas. I named the main array: BalancesSummary.
5.8 Worksheet Rpt Summary

This section’s purpose is to present each budget account’s budget totals, actual totals, period totals, and running totals. It uses one component: BXR.ReportGroupSummaryλ. The complete formula can be seen in the formula box.

Below are the results after entering this formula.
5.9 Worksheet Rpt Detail

This section’s purpose is to present each budget account’s budget items, budget totals, actual totals, period totals, and running totals. It uses one component: BXR.ReportGroupDetailOffsetλ. The complete formula can be seen in the formula box.

Below are the results after entering this formula.

5.10 Section Summary
This sample model provides a solution to a common problem: Departmental budgeting. It is flexible enough to satisfy account-level budgets, zero-based budgets, and rolling budgets. It can, without modeler intervention, accommodate any number of accounts, budget items, actual expenditures, and periods.

It uses less than a dozen distinct components that cover large model sections. Some components are specialized, such as the two reports. Some are general and can be used in many models and several times within the same model, such as the summing and timeline components. Because it uses so few formulas, it can be assembled in under 10 minutes.

6 SUMMARY

We can deliver projects far quicker by assembling models from components rather than writing formulas from scratch. We can deliver models that do things we do not know how to do by leveraging components built by experts. We can enforce corporate business rules by encapsulating business logic into pre-built, simple-to-use components. We can deliver models with fewer errors. This is why CBSE was developed decades ago which is now possible to implement in Excel with LAMBDA.

Creating complex LAMBDA scripts requires more skill; however, using CBSE components requires only remedial Excel skills [Microsoft 2021].
REFERENCES


Brown, Alan (First Edition) 2000. “Large-Scale Component-Based Development”, pages 72, 73, and 95.


Melo, Walcelio and Briand, Lionel 1995. Measuring the Impact of Reuse on Quality and Productivity in Object-Oriented Systems, pages 8, 11 and 14


APPENDIX

7 QUESTIONS AND ANSWERS

Q: How do we get the Advanced Formula Environment (AFE) add-in?

A: From Excel’s ribbon click **Insert > Get Add-Ins**. Search for **Excel Labs**. Click **Add** next to **Excel Labs, a Microsoft Garage project**, then **Continue**. The add-in is now part of Excel and is available to all workbooks.

Q: Where is the AFI add-in?

A: The add-in is located on Excel’s **Home** tab under the **Excel Labs** icon on the far right. Click **Excel Labs** icon then **Advanced Formula Environment**.

Q: Do we have to distribute the Advanced Formula Environment (AFE) add-in with our workbooks?

A: No. AFE is only useful for importing and creating LAMBDAs. It is not needed to execute LAMBDAs.

Q: Where are LAMBDAs stored?

A: LAMBDA names, formulas, and descriptions are stored in Name Manager. LAMBDAs imported from GitHub Gist using AFE have their complete GitHub Gist source stored in the workbook’s XML.

Q: Can I change an imported LAMBDA?

A: Yes. We can change LAMBDAs inside Name Manager, in AFE, in Excel, or even with Notepad.

Q: What happens to my workbook if a new version of a LAMBDA is released on GitHub Gist?

A: Nothing. The old version stays in our workbook. To get the updated version we must delete the old module and download the new version.

Q: Why would a LAMBDA creator come out with new versions?

A: They may have found a bug and fixed it, a more efficient formula, updated documentation, or added a new feature.
Q: If a new feature is added to a LAMBDA, can it break our workbooks?

A. Not if we don’t import it and not if done correctly. As long as the original parameters remain in their positions, and the new features are accessed as optional parameters added to the end of the original parameters, our existing workbooks should be able to import the new versions with no issues.

Q: If a LAMBDA is updated, can we see what changed?

A. Yes if stored on GitHub Gist which has a very robust version control system.

Q: Are LAMBDAAs black boxes? Should we avoid black boxes?

A: No. They are not black boxes and black boxes should not be avoided. SUM() is a black box as are all Excel functions because we cannot see their source. Even so, we use them because we know what they do and we know they work well. LAMBDAs, on the other hand, are not black boxes. We can see their formulas in Name Manager, or, more easily, in AFE where we can see imported LAMBDA documentation.

Q: Do LAMBDAs make models non-transparent?

A: If we mean not viewable from Excel’s interface? No. LAMBDA formulas are viewable in Name Manager and AFE.

Q: Are LAMBDAs harder to understand?

A: That depends on our skill level and that is why LAMBDAs are so valuable. LAMBDAs can give those with less skill the power to do what those with great skill can do. If our skill level is high enough, we can read LAMBDAs just like normal Excel formulas because they are made from normal Excel formulas. And unlike normal Excel formulas, we can name them to convey intent. We can also add documentation IN the LAMBDA. If we use LET, we can break large formulas into smaller sections and name them. Named formulas, smaller named segments, and documentation greatly increase understandability.

Q: If we use a LAMBDA but do not understand how it works, how do we know it is okay to use?

A. We should test it.
Excel as a Turing-complete Functional Programming Environment

Peter Bartholomew
MDAO Technologies Ltd
peter.bartholomew1@btinternet.com

ABSTRACT

Since the calculation engine of Excel was the subject of a major upgrade to accommodate Dynamic Arrays in 2018 there has been a series of seismic changes to the art of building spreadsheet solutions. This paper will show the ad-hoc end user practices of traditional spreadsheets can be replaced by radically different approaches that have far more in common with formal programming. It is too early to guess the extent to which the new functionality will be adopted by the business and engineering communities and the impact that may have upon risk. Nevertheless, some trends are emerging from pioneering work within the Excel community which we will discuss here.

1 INTRODUCTION

Until recently, spreadsheets have very much been the territory of the non-programmer. The majority of spreadsheets in use today do not even contain formulas. In fact, if one is to accept a remark made by David Benaim at a London Excel meetup, the practice of merging cells to assist the presentation of data is more widely encountered than the use of formulas.

When it comes to formulas, the current state of the art is still much as described by [Hellman, 2001] in his EuSpRIG paper, in which he blames most of the recognised deficiencies, such as the profusion of errors, auditing difficulties, lack of uniform development methodologies, and barriers to easy comprehension of the underlying business models, upon the standard spreadsheet user-interaction paradigm of the ‘cell-matrix’ approach. His solution was to develop an interface program that would accept formulas written in natural language and then generate the necessary spreadsheet formulas without further user intervention.

In this paper we will consider how equivalent development strategies are now possible working directly with built-in Excel formulas. The Excel developments that will be described start with the introduction of dynamic arrays, which were described by the present author at EuSpRIG [Bartholomew, 2019], then will proceed to address the impact of the LET function, as found in other functional programming languages, which allows the introduction of locally-scoped named variables to Excel. This was followed by Lambda which permits the development of user-defined functions without any need to resort to VBA or other Add-Ins. From the recording of the online POPL 2021 Conference presentation that featured Simon Peyton Jones, Andy Gordon, Jack Williams of Microsoft Research, it appears that the idea of turning the humble spreadsheet into a full-fledged programming language had been around for the best part of two decades before any implementation was undertaken, [Peyton Jones, 2021]. The claim of Turing completeness will not be demonstrated here but the implication is that it is now possible to program significant software algorithms just by using Excel spreadsheet formulas.

Finally, the paper will look at limitations of the current Excel implementation both in terms of functionality and the testing and verification that is essential to the management of risk.
2 THE EVOLVING SPREADSHEET LANDSCAPE

2.1 The traditional spreadsheet experience

Much of what we consider to be ‘normal’ spreadsheet practice dates all the way back to 1980s and ideas for the electronic spreadsheet developed within VisiCalc by Dan Bricklin and Bob Frankston. The idea of cells within a grid was inherited from manual paper-based spreadsheets but the A1 town plan notation for referencing cells was adopted simply to avoid the ‘tedium’ of the user having to define their terms using named variables. The aim was to allow users to work with a layout that was to be used for the final presentation of results which echoed the development of WYSIWYG word processing interfaces that had emerged only a little time earlier.

The early decisions very much saw the spreadsheet as an environment in which non-programmers could manipulate and present numerical data. That mindset largely persists up to this day. John Raffensperger [Raffensperger, 2001] argued that the idea of “the spreadsheet as a computer program” needs to be replaced and instead it should be viewed as a document that is intended to be read in order to communicate business logic and content. Much of his argument relating to layout are sound, especially in the prevailing context of the single-cell calculations based upon the concept of direct cell referencing.

Where I would disagree with Raffensperger in that “the spreadsheet as a computer program” is not a metaphor; it is the reality. To be in denial does not make the spreadsheet any less a program; it simply increases the likelihood of it being an extremely poorly constructed program. Despite the fact that arrays have been available from the time Excel was ported to Windows, years of experience answering questions posted to on-line forums would suggest that they are viewed as methods of last resort rather than a simplifying assumption that captures the significance of the calculation. Relatively few users have experience of array calculations, possibly because they are not discoverable; no one would stumble across the strategy of committing formulas by pressing Enter, whilst Control and Shift are held down, by accident! The traditional way of performing what should be an array calculation requires the introduction of the concept of relative references, so that a separate formula may be created for each element of the array by using copy/fill.

Again, defined names existed within past versions of Excel, but they are used relatively rarely and some standards like FAST [FAST, 2015] deprecate their use. Without defined names, formulas tend to read more like encrypted cyphers than natural language, so there is pressure to keep them short and ensure that they reference nearby cells. To find the significance of a reference, one must follow the precedence and then hope to find nearby annotation. In his 2016 paper, [Bartholomew, 2016] showed that it is possible to build an entire spreadsheet model using only CSE array formulas and how that, in turn, enabled the use of defined names, since one no longer needs to reference individual cells. As was pointed out during discussion, using techniques that were so far from normal practice would do little to address the problem of spreadsheet error because only a few developers would be in a position to deploy them.

Overall, the assumptions and techniques of the traditional spreadsheet have become so familiar that few would even recognise how much of an outlier spreadsheets are in the world of computing. There is no expectation that variables should be defined; formula references are specified solely by ‘point and click’ and provide no clue as to their business significance; the size of any array depends solely on how far the developer happened to drag the formula when setting it up. With modern Excel, functional programming concepts have entered the delightfully chaotic world of the spreadsheet.
2.2 Dynamic Array formulas

In September 2018 Microsoft announced the impending release of dynamic arrays within Excel. Prior to that, the value property of a single cell was limited to a single value, be it a String, a Double, a Boolean or an Error. At the Florida Ignite meeting Joe McDaid of Microsoft described how the newest releases of Excel could associate an array object with a single cell. Adjacent empty cells, known as a spill Range, would be used to display the result. The size of the spill range depends solely on the formula references, rather than user action, resulting in one element of risk being eliminated as a result.

An interesting fact that emerged at the time is that the calculation engine of Excel always was capable of calculation based upon the use of 2D arrays. It was just on the grid that a function that expected a scalar parameter but was given a vector, e.g.

\[ \text{MOD}(A1:A10, 3) \]

would first reduce the range reference to a single cell by implicit intersection and then return a single value. Had the same formula been specified within a defined name it would have been evaluated as an array formula. Similarly, if the range were converted to an array constant

\[ \text{MOD}((1;2;3;4;5;6;7;8;9;10), 3) \]

the evaluation would have been correct. All that Ctrl/Shift/Enter did was remove the implicit intersection step. In modern Excel the default behavior is reversed and, to force Excel to perform the implicit intersection trick, one would need to use the ‘@’ operator

\[ \text{MOD}(@A1:A10, 3) \]

At the same time as these changes were introduced, several new array functions, including SORT and FILTER, were released. In legacy versions of EXCEL such operations required manual operations using ribbon commands but now the functions could be built into the automatic calculation.

Many spreadsheet users are not attracted to array formulas because they see anything other than scalar arithmetic as undesirably complicated. I would contend that the reverse is true; array formulas create amazingly simple solutions that structure the solution and replace a multitude of single cell formulas with a single readable formula. A hopeful sign is that Microsoft has reported a substantial upturn in the use of array formulas since the release of dynamic arrays. I now see posts that request dynamic range solutions rather than fill-down.

2.3 The LET function

Whereas, for me, the dynamic array formula had been long anticipated, the introduction of the LET function came as a complete surprise. In functional programming environments, LET introduces the scope in which local variables may be defined. The implementation in Excel is managed by having \(2n+1\) parameters in pairs, with odd numbered parameters being the variable names and even numbered parameters the formula that is to be evaluated.

It appears that programmers with greater experience of traditional Excel methods are likely to write the resulting formula as concisely as possible,

\[ = \text{LET}(x,C5:C15=C3,y,FILTER(D5:D15,x),\text{IF}(y<>"",y,"")) \]

whilst those who have embraced the new methods more completely may be more inclined to use a vertical layout

\[ = \text{LET}(\text{criterion, account=required, selected, FILTER(completionDate, criterion)}, \ldots) \]
IF(selected<>"", selected, 

where Alt/Enter (linefeed) is used to emphasise the code nature of the formula as a sequence of statements. Something particularly worthy of note is that the names in a LET statement are variables. That is, they are evaluated when first encountered and the value may then be used multiple times without recalculation. That is to be contrasted with the defined name which refers to a formula that is re-evaluated every time it is encountered.

2.4 The LAMBDA function and recursion

The next step of introducing the LAMBDA function has far more profound implications but, apparently, it was reasonably easy to implement LAMBDA once LET was up and going. Rather than enabling the evaluation of variables within the formula, LAMBDA associates variable names with the terms of a list of function arguments and then passes the values by name into the body of the formula. To illustrate this, one might consider a simple exponential growth calculation throughout a period of time

= ExponentialGrowth(10000, 5%, 12)

where the function is defined with the initial amount, growth per period and number of periods as parameters. The function itself is given by

ExponentialGrowth = LAMBDA(initial, rate, nPeriods,

LET(

    periods, SEQUENCE(1 + nPeriods, , 0),
    initial * (1 + rate) ^ periods
)
)

The formula serves to demonstrate the passing of parameters and, also, the manner in which the naming of Lambda functions can be used to express the purpose of the formula, whilst concealing the details of its calculation. Used well, this allows the developer considerable freedom to refactor the solution without the risk of generating errors as a result.

A more interesting problem arises when the growth factor changes from period to period, for example, as it would for a variable rate mortgage. One way of addressing that problem is through the use of recursion.

2.5 Recursion

Apparently, it is the use of recursion that has made modern Excel Turing complete; that is, capable of simulating the results of any other computer program. Whilst that may be true, it does not mean that recursion is easy to use, especially so given the skill profile of the typical spreadsheet user.

The worksheet formula

= Recur(initialAmount, vRate)

uses a recursive Lambda function
Recur:\n= \text{LAMBDA}(\text{opening}, \text{vRate}, [p], \\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\..
This is easily done by a traditional fill-down formula

\[ \text{SUM}($F10:$G10) \]

but, whilst the table is a dynamic array, a totals column calculated this way is not dynamic. There are dynamic array formulae that use \texttt{MMULT}, but the best solution is to use \texttt{BYROW}

\[ \text{BYROW(return\#, LAMBDA(x, SUM(x)))} \]

which feeds each row of investment returns into the \texttt{LAMBDA} function in turn. It may also be noted that the expression highlighted in red could be converted to a named Lambda function \texttt{Sum\_l} (say), so enabling the calculation to be expressed more concisely as

\[ \text{BYROW(return\#, Sum\_l)}. \]

A more severe problem with dynamic array calculation arose with the task of creating running totals by accumulating flow variables. This is frequently referred to as a ‘corkscrew’ calculation. The problem is that any attempt to reference a previous term in a dynamic array returns a circular reference error.

\[ \text{SCAN runs along a timeline and by defining and passing the values referenced by the second parameter to the Lambda function, Add\_l, one at a time, where it is combined with the result from the previous step to give the new result. The formula used to accumulate the net cash flow is} \]

\[ \text{SCAN(0, Revenue-COGS, Add\_l)} \]

where \texttt{Add\_l} is defined by

\[ \text{LAMBDA(x, y, x + y)} \]

At each step, \texttt{SCAN} will provide the previous result as \(x\) and the current flow value as \(y\), which the function \texttt{Add\_l} then sums.

The Lambda helper functions perform calculations that would otherwise require recursion but with a far more straightforward syntax. There are still shortcomings that prevent multiple accumulations to be performed simultaneously but, even there, workarounds involving \texttt{REDUCE} are possible.
2.7 Array shaping functions

The latest batch of functions are the array shaping functions. The traditional single-cell, manual approaches offer complete flexibility when it comes to the layout selected for the presentation of results, but array formulas are far more constrained. Two of the functions are used in the example showing forecast investment returns in the previous sub-section.

Figure 1, above, shows annual returns for two investments, but also shows a total value for each investment. At first sight one might expect this to generate spill errors as the number of years increases.

\[
\text{LET}(
    \text{escalatedValue}, \text{DROP(value#, -1)} \times \text{escalation},
    \text{investmentTotal}, \text{BYCOL(escalatedValue, Sum)},
    \text{blankRow}, \{"",""},
    \text{VSTACK}(
        \text{blankRow},
        \text{escalatedValue},
        \text{blankRow},
        \text{investmentTotal}
    )
)\]

The functions to note are \text{DROP} which removes the terminal values because only period opening values are used in the calculation of percentage growth. The overall return from each investment is calculated using the \text{BYCOL} function. Finally, the presentation is built using \text{VSTACK} with blank rows used to align and space the table of results.

These can do far more than profile output within the presentation layer. They can have a key role in the calculation process. For example, if you had a row of historical data by quarter, you could examine it for seasonality by wrapping the data by year and averaging over the years.

\[
\text{LET}(
    \text{salesArray}, \text{WRAPROWS(DROP(sales#,1), 4)},
    \text{BYCOL(salesArray / SUM(salesArray), Sum)}
)\]

There are many other things that could be said about Lambda functions, including the fact that they are first class objects within Excel and can be passed as parameters to other Lambda functions. Unfortunately, providing a definitive treatment of Lambda lies beyond the scope of this paper.

What follows instead is a treatment of the types of Lambda usage that has emerged over the past two years and the needs they are intended to meet.
4 BUILDING SOLUTIONS USING LAMBDA FUNCTIONS

4.1 Categories of Lambda function

The initial focus when Lambda was introduced within Excel was on the opportunity it offered for the creation of UDFs without leaving the formula environment. Since then, a number of markedly different categories of Lambda have emerged. These include

- **Simple uses that turn built-in functions into Lambda functions for use with helper functions.** These would include the `Sumλ` and `Addλ` functions used in the previous section. Another possibility might be to modify a built-in function, e.g. to return the sequence number of a record relative to a table, based upon the `ROW` function

\[
\text{Rowλ(record, table)}
\]

- **Functions that are problem-specific and employed to describe the process and intent of a particular solution.** An example might be a function that distributes several recurring payments along a timeline

\[
\text{LAMBDA(start, occurrences, periodicity, amount, counter)} = \text{LET(outflow, SEQUENCE(occurrences, , start, periodicity), BYCOL(IF(counter=outflow, amount), SUMλ))}
\]

where the names are based on the specific terms used within the workbook.

- **Lambda functions that extend the library of in-built functions of Excel by using published Lambdas.** Examples of these are given in the next section, but can include complicated mathematical calculations such as optimisation.

- **Recently Craig Hatmaker [Hatmaker, 2023] has suggested that Lambdas could be used to generate major software components as the building blocks of a given class of related applications.** The payoff would be both in terms of speed when building a new solution and the confidence provided by using pre-tested components.

Having described the plethora of change that Excel has been subjected to over the past few years, it is probably worth pausing to study some applications, since they bear little resemblance to anything that has preceded them.

4.2 Overhead crane motion using 4th order Runga-Kutta method

Although, in the literature, one more often sees examples of financial models than engineering models, over 85% of manufacturing OEMs and their tier 1 supply chain companies use Excel for calculations related to their products. The modelling in this example was intended to demonstrate the ability of Excel to perform highly mathematical calculations, given that claims are made as to its Turing completeness.

The problem considers the motion of a freight container under the action of control forces applied by the operator to the crane cart. The system of equations to be solved are

\[
\begin{align*}
\dot{y} & = v \\
\dot{\theta} & = q \\
\dot{v} & = e\theta + u \\
\dot{q} & = -\theta - u
\end{align*}
\]

where \( y \) and \( v \) are the linear position and velocity of the cart, whilst \( \theta \) and \( q \) are angular measures. The control input is denoted by \( u \).
The worksheet formula used for this is

\[ = \text{SCANV}(X_0, t, \text{RK4Step\(\lambda\)(D\(\lambda\)))} \]

where

\[ \text{RK4Step\(\lambda\)} = \text{LAMBDA}(D, \text{LAMBDA}(x_r, t_r, \text{LET}(
\begin{align*}
\delta x_1, & \quad \delta t \ast D(x_r, t_r), \\
\delta x_2, & \quad \delta t \ast D(x_r + \delta x_1 / 2, t_r + \delta t / 2), \\
\delta x_3, & \quad \delta t \ast D(x_r + \delta x_2 / 2, t_r + \delta t / 2), \\
\delta x_4, & \quad \delta t \ast D(x_r + \delta x_3, t_r + \delta t), \\
x_{r+1}, & \quad x_r + (\delta x_1 + 2 \ast \delta x_2 + 2 \ast \delta x_3 + \delta x_4) / 6,
\end{align*}
\
\)) \]

It is likely that the normal spreadsheet user will not recognise the formula as having anything to do with Excel. The first step is to recognise the syntax of the LET function, with the first argument being a variable name and the next the formula is to be evaluated. The next point is that the names themselves look unfamiliar to an English speaker, though maybe less so to a Greek, but the Unicode characters can be accessed using Insert/Symbol and they form valid Excel Names. The names are constructed to look like the mathematical variables they implement, but one could equally use names like ‘variables3rdTrial’ if that makes one happier.

A further point to note is that SCANV is a Lambda function, written by the author, to generalise SCAN to output an array of arrays. RK4Step\(\lambda\) is the implementation of the 4th-order Runge-Kutta algorithm shown above, but it is itself passed as a parameter to SCANV and accepts a Lambda function D\(\lambda\) as a parameter. D\(\lambda\) implements the dynamic equations, above, and are specific to the problem of the overhead crane.

The purpose of this section is to demonstrate that significant programming exercises may be conducted within the worksheet formula environment; it is not necessary to resort to VBA or JavaScript, and the worksheet formula environment is fast. Before leaving the section, it may be worth looking at the results obtained.
The loading sequence specified had been 2 sec accelerating the cart, followed by 2 sec decelerating at half the intensity to allow the container to swing forward to catch up with the cart. The loading was then reversed, with 2 sec acceleration at half intensity before being brought to rest with a 2 sec deceleration at full intensity.

From the figure, below, it may be seen that this control loading sequence leads to substantial residual oscillations, making it difficult to set the container down at the end of the movement. The system was then optimised using Solver to improve the predicted behaviour.

It turned out that one parameter was sufficient to reduce the residual energy of the system to zero, so the durations of the load application was set, and the optimisation carried out with respect to the load intensity of the 2nd and 3rd control inputs.

![Figure 4. Movement of crane and container before and after optimisation](image)

The solution corresponding to the 2 sec timing for control inputs set the reduced loads to 17.5% of the main loads applied. The key finding of the work, though, was not the solution; it was that Excel now offers a practical method of solving such problems without needing to move to commercial dynamic simulation software.

### 4.3 ModelOff 2013 Challenge Problem

The second use case for modern Excel is a cash flow forecasting problem drawn from Round 2 of the 2013 ModelOff competition and made available by Diamuid Early [Early, 2023] with a recorded presentation of the solution he would build. What is shown here is the same problem refactored to use dynamic arrays and Lambda functions.

The original solution was presented as an assumptions sheet provided by the competition organisers, followed by the calculations as developed by Early. The forecast was presented as a number of blocks covering different aspects of the problem. The blocks included:

- Timing
- Income statement
- Debt schedule
- Working capital – receivables
- Working capital – payables
- Cash Flow

A characteristic of the refactored solution is that rather than having formulas for individual cells or even line items, there are six main functions, each returning an entire block of the financial model. The function name is chosen to capture the intent of the block and the arguments provide a comprehensive list of references.
= TimingBlock(\text{modelStart}, \text{modelDuration})
= IncomeStatement(\text{unitVolume}, \text{unitPrice}, \text{unitDirectCosts}, \text{depreciationMonthly}, \text{interest})
= DebtSchedule(\text{modelStart}, \text{modelDuration}, \text{debtAmortisationDate}, \text{debtFacilityA}, \text{debtAmortisationAmount}, \text{interestRate / 12})
= WorkingCapital(\text{revenue}, \text{cashReceiptTiming})
= WorkingCapital(-\text{costs}, \text{cashPaymentTiming})
= CashFlow(\text{modelDuration}, \text{openingAccReceivable}, \text{openingCashReceiptTiming}, \text{AccountsReceivable#}, \text{openingAccountsPayable}, \text{openingCashPaymentTiming}, \text{AccountsPayable#}, \text{interest}, \text{repayment}, \text{openingCash})

The first function \text{TimingBlock} requires a start date and the number of years the model is to run for. The elements of the timing block form a reasonably conventional pattern with a counter row and monthly periods characterized by opening and closing dates. Slightly more problem specific was the decision to include year and quarter values for each period.

\text{TimingBlock} = \text{LET}(
\text{counter}, \text{SEQUENCE}(1, 12 * \text{modelDuration}),
\text{periodEnd}, \text{EOMONTH}(\text{modelStart}, \text{counter} - 1),
\text{periodStart}, 1 + \text{EOMONTH}(\text{+periodEnd}, -1),
\text{year}, \text{YEAR}(\text{periodEnd}),
\text{quarter}, 1 + \text{QUOTIENT}(\text{MONTH}(\text{periodEnd}) - 1, 3),
\text{VSTACK}(\text{counter}, \text{periodStart}, \text{periodEnd}, \text{year}, \text{quarter})
)

By using \text{VSTACK} to output the counter along with period start and ends, as well as specific data such as year and quarter, the various detailed formulas are brought together in a way that is easily read and checked and the number of formula cells is minimised; the smaller the count of formula cells, the less is the opportunity for error.
It is also possible to generalise such formulas further in order to turn them into generic library functions by including steering data to select the output options and the unit of time represented by a period. In the present case, however, application-specific functionality was sufficient.

The next area of particular interest was the working capital block. The first point of note is that both the accounts receivable and accounts payable were summarised by the same Lambda function.

```excel
= LET(
    cashChange,   (-1) *TAKE(
        Convolveλ(timing, amounts),
        12 * modelDuration),
    closing,    SCAN(0, amounts + cashChange, Addλ),
    opening,    closing - (amounts + cashChange),
    VSTACK(opening, amounts, cashChange, closing)
)
```

The key to the formula is the imported Lambda function `Convolveλ` that generates a convolution of two arrays. Convolutions are used in satellite signal processing to support GPS, image processing and many other high-tech applications. To maintain efficiency for extremely large arrays of data the function is based upon Fast Fourier Transforms. The content is the preserve of professional mathematicians and electrical engineers and would normally be taught over the course of several specialist university lectures.

Despite the complexity of the implementation, the idea of a convolution is relatively simple when applied to small problems such as the present case. Basically, all that is required is to write the two arrays, one as a row and the other as a column and take the outer product of the two arrays. The result sought is the sum of terms arising on the reverse diagonals, as shown in the figure below.

![Figure 6. Calculating convolutions arising from payment schedules or depreciation.](image)

The point of the example is that it is perfectly possible to use imported Lambda functions as black boxes; there is no need to understand the details of the method provided the function has been thoroughly tested and sampling the result shows it to give the correct results.

5 ELEMENTS OF RISK

An objective of this paper is to demonstrate that, using Excel 365, it is now possible to create solutions to problems that have little or no resemblance to past spreadsheet solutions. The approaches are so different that it would not be reasonable to assume that the risks are in any way similar. Whereas traditional spreadsheet solutions are highly interactive processes based on the cell-matrix paradigm, modern dynamic array solutions can have far more in common with formal programming. The only thing that still distinguishes a spreadsheet solution is its use of a 2D grid in which multidimensional array calculations can be performed and presented to the reader.
With these changes comes another problem. It is likely that solutions such as those presented here will be beyond the capabilities of the majority of end-users; they will have neither the time nor the inclination to attempt programming tasks of this nature. It is likely that applications built by developers will coexist with *ad-hoc* workbooks developed along traditional lines.

For traditional spreadsheets the risk is most likely covered by “if you do what you have always done, you will get what you always have” or, in other words, “spreadsheets that have errors like dogs have fleas”. The more formal functional programming approach may well reduce the error rate for spreadsheets but, at present, there is little to no experience so that remains pure speculation. Nevertheless, if we consider Ray Panko’s 2010 taxonomy of spreadsheet errors, one may hazard reasonable guesses as to what aspects of error creation should improve and where the risk might actually increase.

If we consider a development model in which the intended end user specifies the problem they wish to address and a separate developer implements the solution, then either the end user has to have reasonable expectations in terms of spreadsheet domain knowledge, or the developer must have significant application domain knowledge. Any gaps present a risk.

![Figure 6. Taxonomy of Spreadsheet Errors](image)

One thing that may help a modern Excel solution is the extent to which the use of named variables within the **LET** functions capture the intent of the solution rather than simply the actions of the author when creating the model. This should provide an element of built-in documentation to offset the fact that a modern solution can require the use of advanced programming concepts.

Another aspect of error reduction is testing. Excel, as yet, does not support key aspects of testing such as stepping through the code to check each line of code. Nevertheless, it should be possible to use unit testing to build confidence as one goes.

Something that does change, is the role of the end user. It is not realistic to expect them to perform cell by cell formula checks as they might have attempted in the past. The process was based upon the concept of code as a ‘white box’ and such checks served to improve confidence. Unfortunately, experience shows that such checks fail to find a majority of errors, so the additional confidence is largely unjustified. What the end user should do is focus on validation and use simple datasets constructed to demonstrate the code’s fitness for purpose. The end user is the domain expert and it they who are better placed to determine the acceptability of any derived results. Conversely it is the developer that is responsible for line by line checks and unit testing as code verification activities.
As for execution errors, problems of typing and pointing should almost be eliminated. The array solutions do not allow errors here and there; either the result is correct everywhere or nothing is correct. In addition, reducing the number of formula cells from 6,000 to 10 (say) dramatically reduces the opportunity for random error.

In the author’s experience, it is almost inevitable that a developer-built solution will exhibit ‘carbuncles’ after a period of use. As with any application, commercial or bespoke, the end user will from time to time think of additional requirements. It is likely that these will be implemented in the form of traditional formulas making direct reference to cells within the installed workbook. The changes will work and most likely give correct results as they are implemented, but they introduce the risk of latent error. The chances are that such extra code will not respond dynamically to new input data, so the code should be refactored to incorporate new requirements soon after they arise.

6 CONCLUSIONS

Modern Excel claims to be Turing complete and this paper has set out to demonstrate a range of techniques that a world away from the naive constructs, characteristic of traditional spreadsheet solutions. Although the techniques are far more advanced, they capture the intent of the programmer and allow solutions to be developed that would otherwise require VBA or Typescript. It is argued that semantically meaningful coding practices could give more reliable results.

REFERENCES


Bricklin D, “Meet the inventor of the electronic spreadsheet”, https://youtu.be/YDvbDiJZpy0


Hatmaker C, “Reducing Errors in Excel Models with Component-Based Software Engineering”, EuSpRIG 2023 Conference


Bisseling R H, “Utrecht University Lecture, Non-recursive Fast Fourier Transform” https://www.youtube.com/watch?v=WcCl4q2pKcM


Panko R, “What we don’t know about Spreadsheet Errors today: The facts, why we don’t believe them, and what we need to do”, Proceedings of the EuSpRIG 2015 Conference “Spreadsheet Risk Management”
ChatGPT and Excel – trust, but verify
Patrick O’Beirne
Systems Modelling Ltd
Ireland
pob@sysmod.com

ABSTRACT
This paper adopts a critical approach to ChatGPT, showing how its huge reach makes it a useful tool for people with simple requirements but a bad, even misleading guide to those with more complex problems which are more rarely present in the training data and even more rarely have straightforward solutions.

It works through four exercises in creating lookup formulas using chatbots, showing the need to test the offered solutions. They are a simple lookup, a lookup to the left, a match of two values at the same time, and intentionally confusing the models by using common language with technical meaning in Excel.

It concludes with a practical guide for how to add an Excelscript button, with system and user prompts, to the ChatGPT API into the Excel desktop environment, supported by a blog post giving the technical details for those interested.

1 OVERVIEW
We review the subject of Spreadsheet Risk and introduce the notion of Large Language Models (LLM’s). They are attracting increasing attention in multiple domains and discuss the current state of the art and the use of a particular LLM in the field of accountancy, where spreadsheets have a dominant application. We discuss Excel formulas and VBA through a series of video links and related successes and failures together with Security concerns. We show how to add a ChatGPT query box to Excel. We conclude with Ten lessons learned and a summary. In the Appendix we document four exercises creating lookup formulas using chatbots.

1.1 Trust issues in spreadsheets

Spreadsheets are everywhere, used from quick scratchpads to planning decisions, operational processes, valuations, financial reporting, and analysis. Excel is the “Swiss Army knife” for data analysis and calculation. But however qualified users are in their own domain, few have received training in Excel to the level they should have to use it efficiently and effectively in their work. Because they don’t know what they don’t know, they rate themselves overconfidently as “proficient in Microsoft Excel”. This gives rise to what has been called the “Sorcerer’s apprentice” syndrome, named from a story where an apprentice magician knew enough to cast a spell, but not enough to stop it when it ran out of control.

The EuSpRIG horror stories page [EuSpRIG, 2020] has many stories of the expensive and embarrassing results of out-of-control spreadsheets, such as the JP Morgan $6 Billion trading loss, and the Reinhart and Rogoff ‘austerity spreadsheet’ error. Data quality issues
are frequently reported, for example the articles in scientific journals about the widespread occurrence of gene name errors caused by Excel misinterpreting names such as SEPT2 as a date. As a result, researchers have now renamed genes to avoid this problem.

A large body of research has been published by the European Spreadsheet Risk Interest Group (EuSpRIG, 2020) on the risks in using spreadsheets, stemming from user under-training but over-confidence. Trust issues in automated solutions have been researched by Muir & Moray [1994] and Croll [2017]. Many online forums exist to support users of Excel and they represent a very large resource of knowledge. Millions of users trawl these every day either spending hours Googling and sifting the useful information from the spam, or posting their current Excel challenges and receiving answers, sometimes choices of answers, from Excel experts such as Microsoft Valued Professionals (MVPs). Now they have new machine-learning (branded "AI") tools with vastly more patience and capacity than they have to scrape the millions of existing questions and answers. These are the large language models (LLMs) such as ChatGPT-4 which are behind ChatGPT, Microsoft Bing and Copilot [Microsoft, 2023], and Google Bard.

1.2 Large Language Models

A Large Language Model (LLM) is an algorithm has been trained on a large amount of text-based data, typically scraped from the open internet, and so includes billions of web pages and - depending on the LLM - other sources such as scientific research, books or social media posts. Recent examples of LLMs include OpenAI's ChatGPT, Microsoft Bing, Google’s Bard and Meta’s LLaMa (for scientific papers).

They are very good at reproducing the syntax of sentences in natural languages but have no concept of semantics, the meaning of the words. But that does not stop people attributing intelligence and understanding to them. Some people are now calling themselves “Prompt Engineers”. Prof. Ethan Mollick of Wharton University has a blog [Mollick, 2023] on how to optimize the use of ChatGPT and other LLMs.

In the domain of spreadsheets, or code more generally, testing is the most important step, because users of ChatGPT necessarily do not have the skills to create the formula themselves, or even to understand how the offered formulas work, so all they can do is test.

1.3 Current state of the art

For common problems with published solutions, the suggestions are correct most of the time. That reflects the weight of such Q&As in the sources. When there is less information to scrape, ChatGPT makes stuff up based on whatever it can find on similar topics. When asked to create a list of the ten most cited papers on a research topic, the first two were close to real papers, the rest were completely invented. I tested both ChatGPT and Bing by giving each the top five papers and asking them to check whether they were genuine by providing links to online sources. ChatGPT invented non-existent links, or links to completely different papers, and created more different but non-existent links when the response was regenerated. Bing gave a link for one, a correct paper, and declined to provide links to the rest, saying that they were either not genuine or behind a paywall.
Bing found two different papers and said that it could not find more. Bing is less creative and more factual. It gives links to search results to learn more. Microsoft have announced Copilot, a formula assistant in Excel [Microsoft, 2023].

In an article on the website of the Institute of Chartered Accountants of England & Wales (ICAEW) "ChatGPT and accountancy: know its limits" [ICAEW, 2023], Stuart Cobbe put it through the ICAEW assurance assessment paper. The version in place at the time, ChatGPT-3.5, failed with a score of 42%. When ChatGPT-4 was released, Cobbe tried the assessment again. This time, it passed with 78% – a significant improvement on last time, with better understanding and reduced rambling. You should consider it, says Cobbe, “like an ill-informed but confident junior.”

1.4 ChatGPT successes

Mynda Treacy [Treacy, 2023] pasted in a question on Power Pivot from her forum, and it gave the correct answer using the column names in the question.

I tried a question from the Excel-L list: Tell me why I get an error "Syntax error in INSERT INTO statement." in the following SQL statement:

INSERT INTO stagingHatchCommList ( Type, Direction, To, ContactFrom,…

Answer: “may be caused by the use of reserved keywords as column names”

“Use INSERT INTO stagingHatchCommList ( [Type], [Direction], [To], …”

Paul Kelly [Kelly, 2023] asked it to create a VBA Class module, a a wrapper class for a Collection that implements a Reverse method. The answer impressed him.

1.5 ChatGPT failures

I asked it for a formula to convert IP addresses with four 3-digit parts that have been misinterpreted as numbers, eg 121451781090, into the correct form, eg 012.145.178.090

ChatGPT offered several solutions, none of which worked.

One solution is SUBSTITUTE(TEXT(A7,"000|000|000|000"),"|",";"); another uses a custom number format, =TEXT(A7,"000\.000\.000\.000").

1.6 Security concerns

The UK National Cyber Security Centre (NCSC) [UK NCC, 2023] reports that “…concerns that the OpenAI tools could turn any would-be petty thief into a technically savvy malicious coder in moments were, and still are, well-founded. ChatGPT-enabled cyberattacks started popping up just after its user-friendly interface premiered in November 2022. … For more complex tasks, it's currently easier for an expert to create the malware from scratch, rather than having to spend time correcting what the LLM has produced. However, an expert capable of creating highly capable malware is likely to be able to coax an LLM into writing capable malware.”

They advise:

Do not include sensitive information in queries to public LLMs

Do not submit queries to public LLMs that would lead to issues were they made public
Add a ChatGPT-3.5 query box to Excel

The Author’s blog [O’Beirne, 2023] shows a way of building an interface to the OpenAI ChatGPT-3.5 model.

It uses Excelscript (a variety of Typescript) so sits in your Excel desktop application, not in a workbook, so it does not need VBA. (VBA macros can be found as well, of course [Mullaned 2002, 2023])

You can download the ChatGPT in Excel demo workbook from [O’Beirne, 2023]

2 SUMMARY

2.1 Lessons Learned

The ten lessons learned from the experiments in the Appendix are:

1: Test the formula suggested, even if it looks right to you – or in other words, you don't see any problem at first.

2: If it does not work, try again. Give it more context, say what does not work.

3: Start over again with a fresh session and ask the same question to see if you get the same answer.

4: If you know of a newer function that you think might work, suggest it in your prompt.

5: When you copy down a formula, ensure that the references are correct in both relative and absolute forms.

6: Do more than one test and verify each result.

7: It's best to really understand how the formula works so that you can decide whether it is a good match for your needs.

8: When you get a formula, ask what could go wrong with it and check each potential issue.

9: When you get a formula you can't understand, ask a human expert to explain it.

10: Remember, these are language models, not logical or mathematical models.

2.2 In conclusion – trust, but verify

Use LLMs for questions where there should be a known solution. Verify any statements against trusted sources. Bing provides links to known sources. Test any code (Excel formulas, VBA, or other programming code) with realistic data and ensure that the results are correct before putting it to use. Repeat and refine the suggestions to see if you get a better answer.
REFERENCES


Mynda Treacy (16 Feb 2023) ChatGPT for Excel - what they’re NOT telling you!. YouTube. https://www.youtube.com/watch?v=89gMzq98LPs&t=2s


APPENDIX

Experiments in ChatGPT and Microsoft Bing with Excel formula questions

In the dialogues following, I have edited the questions and verbose responses to save space. I chose to try lookup exercises.

Exercise 1: a simple lookup

I have a list of products and prices in A10:B110, and a list of the products sold and their sales quantity in J11:K50. In column L I want a calculation of the quantity times the price for each product.

Figure 1: simple lookup example with one formula completed

ChatGPT-3 was correct: =VLOOKUP(J11,$A$10:$B$110,2,FALSE)*K11

The VLOOKUP function examples returned all specified the fourth parameter [range_lookup] correctly as FALSE, which does not require the lookup list to be sorted in ascending order and therefore uses a linear search from the beginning of the list. They did not make the common user mistake of omitting it; the default value for the omitted parameter is TRUE, which requires the lookup list to be sorted so that a fast binary split search can be used.

ChatGPT-3.5 first got the columns the wrong way around: =J11*VLOOKUP(K11,$A$10:$B$110,2,FALSE). When told that, it corrected it. A naive user might just type it in without noticing that and would get a #VALUE! result. Even if they did notice it, they may still decide to do it as a test.

Lesson 1: Test the formula suggested, even if it looks right to you – or in other words, you don't see any problem at first.

Bing made the same mistake: =J11*VLOOKUP(K11,$A$10:$B$110,2,FALSE). When told "The product name is in column J and the quantity in column K", it responded "Thank you for letting me know. …. =K11*VLOOKUP(J11,$A$10:$B$110,2,FALSE)".

Exercise 2: reverse lookup, ie lookup to the left

I then tried a swapping the columns for product name and price, asking for separate formulas for lookup and multiplication.
I have a list of products in C11:C110, and prices in B11:B110, and a list of the products sold in J11:J50 and their sales quantity in K11:K50. In column L I want the price of each product sold and in column M a calculation of the quantity times the price for each product.

![Figure 2: more complex lookup with the lookup column after the returned column](image)

They all got the simple multiplication correct in M12, so I will focus on the lookup formula in L12.

ChatGPT-3.5 was incorrect at first: \(=VLOOKUP(J11,$C$10:$B$110,2,FALSE)\)

This formula is not correct syntax because C is after B. If you entered that in Excel, it would autocorrect it to \(=VLOOKUP(J11,$B$10:$C$110,2,FALSE)\). When I asked "What is wrong with this Excel formula: \(=VLOOKUP(J11,$C$10:$B$110,2,FALSE)\)" it correctly replied "The range \$C\$10:\$B\$110 is in the wrong order. It should be \$B\$10:\$C\$110." Later, I tested more variations on this, see problem #4 below.

Worse than that, the formula does not work because it is looking up the product name in column B which is the product price. Applying lesson 1 and testing that, I get a #N/A value. I therefore revised the prompt by adding "Note that the product prices are to the left of the product names so I cannot use VLOOKUP." It then offered: 
\(=INDEX($B$11:$B$110,MATCH(J11,$C$11:$C$110,0))\) which is correct. Sometimes simply repeating the prompt will get it to offer a different answer, but it's better to be more specific.

**Lesson 2: If it does not work, try again. Give it more context, say what does not work.**

ChatGPT-3.5 gave the same incorrect formula: 
\(=VLOOKUP(J11,$C$10:$B$110,2,FALSE)\). When I replied that that have a #N/A, it responded "I apologize for the mistake. The formula for column L should use the price column as the first argument and the product column as the second argument in the VLOOKUP function. Here's the corrected formula for column L: 
\(VLOOKUP(J11,$B$10:$C$110,2,FALSE)\) This formula looks up the price of the product sold in column J, in the B10:C110 table, and returns the value in the second column (price)." On being told again that this produced a #NA, it offered: 
\(=INDEX($B$10:$B$110,MATCH(J11,$C$10:$C$110,0))\). Repeatedly saying only "that is wrong" is a slow way of being offered an alternative, it is better to provide a reason if you know what it is; which of course naïve "copy & paste" Excel users may not.

Bing made the same VLOOKUP mistake, which I am not re-quoting here, for economy. On the third attempt, it offered for L11: 
\(=INDEX(B10:B110,MATCH(J11,C10:C110,0))\)
This will look up the value in J11 (the product sold) in column C and return the corresponding value from column B (the price). You can then drag this formula down to fill column L.

Unfortunately, when I copy that down, the relative references adjust so in M12 the formula reads =INDEX(B11:B111,MATCH(J12,C11:C111,0)), and so on down, until eventually the answers are #N/A because the references are far out. Would a naïve user think to check the copied cells, and wonder why the #N/A values appear? To know what the problem was, they would have to understand relative and absolute addressing.

In this case, I repeated the prompt in a new Bing topic and it gave the correct formula: "In cell L11, you can enter the formula =INDEX($B$11:$B$110,MATCH(J11,$C$11:$C$110,0))" It also cited four internet sources, which ChatGPT cannot do. For example, "How to use INDEX and MATCH | Exceljet. https://exceljet.net/articles/index-and-match"

Because these are probabilistic language models, they will always suggest the most frequently occurring solutions found in their sources, in this case VLOOKUP, even when they are not appropriate to the specific question. When told not to, they will then fall back to the next most common solution, INDEX/MATCH combination.

I tried telling it to use LOOKUP rather than VLOOKUP. I knew that this was an inappropriate function because LOOKUP requires the lookup range to be sorted in ascending order, but wanted to see what it would do. I expected to get =LOOKUP(J11, $C$11:$C$110, $B$11:$B$110). ChatGPT-3.5 offered the following instructions, to which I have added my comments

<table>
<thead>
<tr>
<th>ChatGPT-3.5</th>
<th>My analysis</th>
</tr>
</thead>
<tbody>
<tr>
<td>Prompt: In an Excel sheet I have a list of all product prices in B11:B110, all product names in C11:C110, and another list of the names of the products sold in J11:J50. Neither list is sorted in any order. In column L I want an Excel formula to return the price of each product sold. Note that the product prices are to the left of the product names so I cannot use VLOOKUP, I want to use LOOKUP.</td>
<td>We don't need to, we could use C as the lookup vector (ie the list to look up) and B as the result vector (ie the list to take the result from)</td>
</tr>
<tr>
<td>Response: To use LOOKUP in this scenario, you need to rearrange the data so that the product names are in the left column and the prices are in the right column. Here are the steps:</td>
<td></td>
</tr>
<tr>
<td>1. Select cells A11:B110.</td>
<td>But the data is in B11:C110</td>
</tr>
<tr>
<td>2. Cut the data (Ctrl+X or right-click and select Cut).</td>
<td></td>
</tr>
<tr>
<td>3. Right-click cell A11 and select Transpose.</td>
<td>There is no Transpose option available after a cut, only after a copy. Assume for a test that we choose Copy and a different output cell temporarily, then</td>
</tr>
<tr>
<td>Step</td>
<td>Action Description</td>
</tr>
<tr>
<td>----------------------------------</td>
<td>---------------------------------------------------------------------------------------------------------</td>
</tr>
<tr>
<td>4.</td>
<td>Right-click cell A11 again and select Paste Special. That transposes the two columns to two rows.</td>
</tr>
<tr>
<td>5.</td>
<td>In the Paste Special dialog box, select Transpose and click OK. That would just repeat the previous paste and would have no effect. If we copied again, chose another destination, and copy &amp; paste transposed, that merely gives us back our original columns, they are not swapped.</td>
</tr>
<tr>
<td>6.</td>
<td>Delete the original data in B11:C110.</td>
</tr>
<tr>
<td>7.</td>
<td>Rename column A to &quot;Product Name&quot; and column B to &quot;Price&quot;.</td>
</tr>
</tbody>
</table>

Now that the data is rearranged, you can use LOOKUP to find the price of each product sold. Here's the formula for cell L11:

\[
\text{=LOOKUP(J11,$A$11:$A$50,$B$11:$B$50)}
\]

The lookup and return areas should be 100, not 50. The LOOKUP function requires the lookup list to be sorted which it is not. ChatGPT doesn't know that, of course as I didn't say so. This formula looks up the value in cell J11 (the product name) in the range A11:A50 (the product names) and returns the corresponding value from the range B11:B50 (the prices). You can copy this formula down to cells L12:L50 to get the prices for all the products sold.

When I tried again, it gave a different method of constructing rearranged columns using INDEX/MATCH, then the same LOOKUP function. INDEX/MATCH would work, but it would be an unnecessary complication.

I also tried explicitly saying that the lists were not sorted and asked if I could use LOOKUP rather than requiring it, as follows:

"In an Excel sheet I have a list of all product prices in B11:B110, all product names in C11:C110, and another list of the names of the products sold in J11:J50. Neither list is sorted in any order. In column L I want an Excel formula to return the price of each product sold. Note that the product prices are to the left of the product names so I cannot use VLOOKUP. Can I use LOOKUP?"

The response from ChatGPT3.5 was that I could, although we know that this cannot work with unsorted lists. At least it did not tell me to rearrange the data:
"Yes, you can use the LOOKUP function to return the price of each product sold. Here's the formula you can use in cell L11: =LOOKUP(J11,C$11:C$110,B$11:B$110)

This formula will search for the product name in cell J11 in the range C11:C110, and return the corresponding price from the range B11:B110. You can then copy this formula down to cells L12:L50 to get the prices for all the products sold."

Bing responded: "No, you cannot use the LOOKUP function in this case because the lookup value must be in the first column of the array. Instead, you can use the INDEX and MATCH functions together to achieve the desired result. In cell L11, you can enter the formula `=INDEX(B11:B110,MATCH(J11,C11:C110,0))`. This is a correct solution for J11, although the reasoning about LOOKUP was wrong. It would not work copied down unless the C and B references were absolute.

I then asked about XLOOKUP. This is a new function in Office 365 and so will appear in fewer web searches and therefore have a low probability of being suggested by a LLM. This worked but no response used absolute references in the lookup table.

ChatGPT-3.5 responded "Yes, you can use XLOOKUP to return the price of each product sold. Here's the formula you can use in cell L11: =XLOOKUP(J11,C11:C110,B11:B110). This is correct for J11 (but not suitable for copying down because not absolute) even though the optional match_mode parameter was not specified, because the XLOOKUP default is an exact match.

Bing gave the same relative-only formula and helpfully added "Please note that XLOOKUP is not available in Excel 2016 and Excel 2019. If you have an older version of Excel, you may need to use a different function such as INDEX and MATCH." It gave links to support.microsoft.com and other sites.

Lesson 3: Start over again with a fresh session and ask the same question to see if you get the same answer.

Lesson 4: If you know of a newer function that you think might work, suggest it in your prompt.

Lesson 5: When you copy down a formula, ensure that the references are correct in both relative and absolute forms.

Exercise 3: Lookup to match two values and return the last match

This is moderately difficult because it requires matching multiple rows in a table and returning the last result that satisfies the criteria. It requires a deeper knowledge of Excel to understand, test, and debug such formulas.

Here is the lookup table, a list of several price changes on different dates for each of three products:
My first prompt did not specifically say that there would be more than one price for a given product:

"I have in Excel a table with product price history from A2 to D10. Column B is the list of products, column C the dates that the product price was changed, and column D is the price set on that date. I want to find the price in column D for a given product name in G2 and a given sales date in H2."

So the first formula that both ChatGPT and Bing offered would only work if there was only one price per product, as it returns the first exact match:

=VLOOKUP(G2, B2:D10, 3, FALSE)

I then specified it more carefully by adding "The sales date can be any date in between two price changes." It still offered VLOOKUP but even more incorrectly:

"To find the price of the product sold on a given date, you can use the following formula:
=VLOOKUP(G2,A2:D10,4,TRUE) This formula uses the VLOOKUP function to search for the product name in column A and return the corresponding price from column D. The "TRUE" argument in the formula tells Excel to find an approximate match for the sales date in column C, which means it will return the price for the most recent date before the sales date."

I tried again, being even more explicit:

"I have in Excel a table with product price history from A2 to D10. Column B is the list of products, column C the dates that the product price was changed, and column D is the price set on that date. Given a product name in G2 and its sales date in H2, I want to find the price in the product history for the given product name in G2 and the latest date not on or after the given date in H2."

Now both ChatGPT (and ExcelFormulaBot.com which uses ChatGPT) and Bing offered an INDEX/MATCH formula.

=INDEX(D2:D10,MATCH(1,(B2:B10=G2)*(C2:C10<=H2),0))

However, that is not correct and their explanation of how it worked was not correct. I have put that in bold font. Because it finds the first 1, it actually returns the first date
before the given date, not the last. I have left the rest of the explanation there for readers
who wish to understand how to lookup an array of calculated Boolean values from
multiple conditions combined by AND.

"This is an array formula, so you need to press Ctrl + Shift + Enter instead of just Enter
after typing it in. Here's how the formula works:

- MATCH(1,(B2:B10=G2)*(C2:C10<=H2),0) returns the row number of the first match
for the given product name in G2 and the latest date not on or after the given date in
H2. The formula uses two conditions: (B2:B10=G2) checks if the product name in each
row matches the given product name in G2, and (C2:C10<=H2) checks if the date in each
row is not on or after the given date in H2. The two conditions are multiplied together
using the * operator, which returns an array of 1's and 0's. The MATCH function then
looks for the first 1 in the array and returns its position.

- INDEX(D2:D10,MATCH(1,(B2:B10=G2)*(C2:C10<=H2),0)) returns the value in
column D at the row number returned by the MATCH function. This gives you the price
for the given product name in G2 and the latest date not on or after the given date in H2."

I tried yet again, adding to the prompt "There can be multiple price change entries for
each product."

Now ChatGPT supplied a formula that checked for the date being the maximum date of
change in the list for the given product and dates on or before the sales date:

=INDEX(D2:D10,MATCH(1,(B2:B10=G2)*(C2:C10<=MAX(IF(B2:B10=G2,C2:C10,H2+1))),1))

This, as is often said, "looks right". And in fact, a first test gives the correct answer for the
Appetizer product. But not for the Granny Smith.

\[
=INDEX(DS2:DS10,MATCH(1, (SB2:SB10=G2)* (SC2:SC10<=MAX(IF(B2:B10=G2,SC2:SC10,SH2+1))))),1)
\]

**Figure 4 – an incorrect solution**

To understand why, you need to see among all those brackets that there is no match_mode
specified for the MATCH function, so it defaults to a binary split search for a 1 in an
array of mixed 0s and 1s.

=MATCH(1,{0;1;0;0;1;0;0;1;0;1;0})

I leave it as an exercise for the reader to manually practice such a binary search in each
case and see how it lands on essentially a random position depending on the unsorted
array.

**Lesson 6: Do more than one test and verify each result.**
Lesson 7: It's best to really understand how the formula works so that you can decide whether it is a good match for your needs.

Those two lessons are perhaps the reason why many spreadsheets created with the assistance of LLMs will be bug-ridden because users who only know how to copy & paste such formulas do not understand how they work.

I asked "What could go wrong with this Excel formula:
=INDEX(D2:D10,MATCH(1,(B2:B10=G2)*(C2:C10<=MAX(IF(B2:B10=G2,C2:C10,H2+1)) ),1)"

ChatGPT said "As an AI language model, I cannot see the data in the cells, so I cannot determine if the formula is correct or not. However, here are some potential issues…” and gave five issues including "3. If the data in the ranges B2:B10 and C2:C10 are not sorted in ascending order, the formula may not return the correct result."

Lesson 8: When you get a formula, ask what could go wrong with it and check each potential issue.

ChatGPT-3.5 first offered an ingenious solution: interpolation. I didn't pursue it.

"To find the price at which the product would have been sold on a given sales date, you need to look up the two dates between which the sales date falls and then use linear interpolation to calculate the price. Here's a step-by-step guide to doing this: Sort the table by the dates in column C in ascending order to make sure that the dates are in chronological order. Use the MATCH function to find the row number of the last date in column C that is less than or equal to the sales date. =MATCH(H2,C:C,1) Use the INDEX function to get the date of the most recent price change before the sales date……". I asked again and got

=INDEX(D:D,MATCH(1,(B:B=G2)*(C:C<=H2),0)+MAX((B:B=G2)*(C:C<=H2)*(C:C))

(B:B=G2)*(C:C<=H2) gives {0;0;1;0;0;1;0;0;0;0} MATCH(1, {0;0;1;0;0;1;0;0;0;0}, 0) returns the first exact match of 1, which is row 3 so that's the FIRST date the product price was changed on; not the last date before the sought date.

((B:B=G2)*(C:C<=H2)*(C:C) returns {#VALUE!;0;43908;0;0;44197;0;0;0;0} because there is a header label row 1. The MAX, intended to find the highest valid date, would therefore return #VALUE! and the only way a user would know why is to perform the array calculation as I did to see where the error value came from.

Even if that did work by removing the header label, adding 3+44197 would give a number 44200 as the INDEX row, which is completely wrong.

By the way, each time ChatGPT says "This is an array formula, so you need to press Ctrl+Shift+Enter instead of just Enter when entering it", that advice may not apply in modern Excel with dynamic array formulas. It doesn't do any harm, though.

Lesson 9: When you get a formula you can't understand, ask a human expert to explain it.
ChatGPT-3.5 gave this formula, with an explanation. It returns the price in the row containing the maximum date that satisfies both the product and date criteria.

=INDEX(D:D,MAX(IF((B:B=G2)*(C:C<=H2),ROW(C:C)),1))

If a range is chosen not starting in row 1, the index number needs to be reduced accordingly. If starting in row 2, take 1 off:

=INDEX(D2:D10,MAX(IF((B2:B10=G2)*(C2:C10<=H2),ROW(C2:C10))-1,1))

So finally, a correct answer is produced. I retried ExcelFormulaBot saying that there were multiple dates for a product, and it offered:

=LOOKUP(2,1/((B2:B10=G2)*(C2:C10<=H2)),D2:D10)

LOOKUP is an old deprecated Excel function but it is still proposed in many Excel forums as a solution because of its behaviour of returning the last match when there are multiple matches. Today we would use XLOOKUP but no assistant proposed that because it has as yet a low frequency in the LLM data sources. But if you ask to use XLOOKUP, they do provide formulas using it. My XLOOKUP formula is


The Boolean multiplications create an array \{0;1;0;0;1;0;0;0;0\}. Because LOOKUP requires the array to be sorted but ignores error values, the zeroes are changed to #DIV/0! by dividing 1 by the values. LOOKUP then just sees a series of 1s and uses the position of the last one found as an index into the prices.

Repeating ChatGPT-3.5 for interest produced yet another version, using AGGREGATE:

=INDEX(D:D,AGGREGATE(14,6,ROW($B$2:$B$10)/($B$2:$B$10=G2)/($C$2:$C$10<=H2),1))

The 14 is the function number for LARGE, 6 means ignore error values, and the rest creates an array of either DIV/0 errors or row numbers where the conditions are met.

Bing at first suggested the incorrect VLOOKUP and INDEX/MATCH formulas described above. It finally offered the LOOKUP(2 version:


Exercise #4: trying confusions with the models

Back in exercise #2, I said that when I asked "What is wrong with this Excel formula: =VLOOKUP(J11,SCS10:SCS110,2,FALSE)" it correctly replied "The range SCS10:SCS110 is in the wrong order. It should be SBS10:SCS110." I wondered how far a probabilistic language model could go in recognising logical or structural inaccuracies. For example, it cannot solve Wordle problems because it works in tokens, fragments of words, rather than letters. I therefore tried it with less common column letters than B or C, which are very common in internet questions and solutions.
<table>
<thead>
<tr>
<th>Question</th>
<th>Answer</th>
<th>Assessment</th>
</tr>
</thead>
<tbody>
<tr>
<td>I first tried columns JAN and FEB. JAN is after FEB in column order so</td>
<td>ChatGPT: The range Jan:Feb is not a valid range reference. It should be a specific range</td>
<td>Incorrect</td>
</tr>
<tr>
<td>if you enter JAN:FEB Excel will autocorrect it to FEB:JAN.</td>
<td>such as A1:B10.</td>
<td></td>
</tr>
<tr>
<td>What is wrong with this Excel formula:</td>
<td>Same</td>
<td></td>
</tr>
<tr>
<td>=VLOOKUP(2,Jan:Feb,2,FALSE)</td>
<td>Bing: There is nothing wrong with the formula itself.</td>
<td>Correct</td>
</tr>
<tr>
<td>Retry with capital letters: What is wrong with this Excel formula:</td>
<td>The formula you provided appears to be a valid VLOOKUP formula. However, without more</td>
<td></td>
</tr>
<tr>
<td>=VLOOKUP(2,JAN:FEB,2,FALSE)</td>
<td>context about the data in the JAN and FEB named ranges…</td>
<td></td>
</tr>
<tr>
<td>I said that JAN &amp; FEB were column letters, not range names. Bing</td>
<td>Bing: In summary, while <code>JAN</code> and <code>FEB</code> may be valid combinations of letters, they are</td>
<td>Incorrect</td>
</tr>
<tr>
<td>insisted that they were incorrect and even after a long exchange</td>
<td>not valid column references in Excel because they do not follow the base-26 numbering</td>
<td></td>
</tr>
<tr>
<td>showing that they were columns 6800 and 4188, and it agreed that JAN</td>
<td>system used by Excel to represent columns.</td>
<td></td>
</tr>
<tr>
<td>was between JAM and JAO it still insisted. You can see why people say</td>
<td></td>
<td></td>
</tr>
<tr>
<td>that Bing's persona is &quot;Fight me&quot;.</td>
<td></td>
<td></td>
</tr>
<tr>
<td>What is wrong with this Excel formula:</td>
<td>The range XYC:XYB is in reverse order.</td>
<td>Correct</td>
</tr>
<tr>
<td>=VLOOKUP(2,XYZ:XYB,2,FALSE)</td>
<td></td>
<td></td>
</tr>
<tr>
<td>What is wrong with this Excel formula:</td>
<td>The range &quot;WAY:WAS&quot; is not a valid range reference. It should be a specific range of cells,</td>
<td>Incorrect</td>
</tr>
<tr>
<td>=VLOOKUP(2,WAY:WAS,2)</td>
<td>such as &quot;A1:B10&quot;.</td>
<td></td>
</tr>
<tr>
<td>What is wrong with this Excel formula:</td>
<td>The range specified in the formula is incorrect. The range should be specified as</td>
<td>It seems to see</td>
</tr>
<tr>
<td>=VLOOKUP(2,ABC:ABB,2)</td>
<td>ABC:ABB instead of ABB:ABC.</td>
<td>that something is</td>
</tr>
<tr>
<td>What is wrong with this Excel formula:</td>
<td>The range specified in the formula is incorrect. The range should be specified as ABC:ABB</td>
<td>wrong but does not</td>
</tr>
<tr>
<td>=VLOOKUP(2,FEB:JAN,2,FALSE)</td>
<td>instead of ABB:ABC.</td>
<td>know what.</td>
</tr>
</tbody>
</table>

Lesson 10: These are language models, not logical or mathematical models.
ABSTRACT

Large Language Models (LLM) have become sophisticated enough that complex computer programs can be created through interpretation of plain English sentences and implemented in a variety of modern languages such as Python, JavaScript, C++ and Spreadsheets. These tools are powerful and relatively accurate and therefore provide broad access to computer programming regardless of the background or knowledge of the individual using them. This paper presents a series of experiments with ChatGPT to explore the tool’s ability to produce valid spreadsheet formulae and related computational outputs in situations where ChatGPT has to deduce, infer and problem solve the answer. The results show that in certain circumstances, ChatGPT can produce correct spreadsheet formulae with correct reasoning, deduction and inference. However, when information is limited, uncertain or the problem is too complex, the accuracy of ChatGPT breaks down as does its ability to reason, infer and deduce. This can also result in false statements and “hallucinations” that all subvert the process of creating spreadsheet formulae.

1.0 INTRODUCTION

1.1 Large Language Models

Large Language Models (LLM) such as ChatGPT or Google Baird are deep learning neural networks trained on vast corpora of human generated language (Jiang, et al., 2020). Over millions of iterations, the neural network builds a probability-based language model of different words it encounters in the corpus. This allows the algorithm to build a detailed probabilistic network of what words relate to one another, how they are used and what they mean in the context of similar terms. When a user inputs a prompt, the text is analysed, the most probable intent or context of the prompt is settled upon, and a corresponding response is provided. ChatGPT optimises 167 billion parameters, making it the largest and in theory the most capable LLM at present.

LLMs can also be leveraged for code generation in a variety of languages including spreadsheet formulae (Chen, et al., 2021). This is based on the same approach for natural language generation, a probabilistic output based on the most likely response to the prompt input by the user. The user inputs plain text prompts that describe and specify what the user wants the code to compute and ChatGPT provides the most likely solution based on the prompt.

1.2 Producing spreadsheet formulae in ChatGPT

Using plain English descriptions, it is possible to create spreadsheet formulae with ChatGPT. If used correctly, this is a powerful tool that could allow those who cannot wield the syntax of Excel or other spreadsheet languages to use the AI generated versions.
For instance, consider a prompt for creating a grading formula for student work at a university:

The grade is based on an average of two cells, if the average is 70 or greater then award a 1st, if the average is equal to 60 and less than 69 then award a 2:1, if the average is equal to 50 and less than 59 award a 2:2, if the average is equal to 40 and less than 49 then award a third, otherwise award a fail.

The above text prompt produces the following accurate spreadsheet formula:

\[
=\text{IF}(\text{AVERAGE}(A1,B1)\geq 70, "1st", \text{IF}(\text{AVERAGE}(A1,B1)\geq 60, "2:1", \text{IF}(\text{AVERAGE}(A1, B1)\geq 50, "2:2", \text{IF}(\text{AVERAGE}(A1, B1)\geq 40, "3rd", "Fail"))))}
\]

However, the quality of the output is directly related to the quality of the input and the more specific the prompt, the more accurate the formulae that ChatGPT creates. If some details are omitted or not specified, the algorithm will omit and take the text literally.

Hence ChatGPT is a top-down approach to coding, one has to understand and describe all parameters of the problem in order to generate valid and useful code. Based on these observations a number of experiments were devised.

1.3 Experiments with ChatGPT

The overall aim of these experiments is to determine how accurate ChatGPT is at producing valid spreadsheet formulae and solving “computational” problems with uncertainty, inference and deduction.

A number of research questions and hypotheses are identified:

**Research Question 1:** How does ChatGPT perform code generation when it is required to solve an incompletely described problem?

**Research Question 2:** What underlying knowledge and competence does ChatGPT have in logic, deduction and inference?

**Hypothesis 1:** The accuracy of solutions provided by ChatGPT will vary depending on the amount of uncertainty in the problem description.

**Hypothesis 2:** ChatGPT has a limited ability to reason with logic, deduction and inference and this limited ability will be obvious when solving more complex logic problems

1.4 Experimental tasks

There are three separate experiments testing the accuracy of ChatGPT. These tasks will use different types of problems to test different aspects of the technology. All experiments are conducted with ChatGPT 4

1.4.1 Experiment 1: problem solving with uncertainty

The experiment task for problem solving is the wall task (Rakovic, et al., 2019; Teo & Tan, 1997; Irons, 2003; Panko & Halverson, 2001; Panko & Sprague, 1998). The wall task is used as a standard way of illustrating how a relatively simple spreadsheet task can generate an array of errors and has been used many times in spreadsheet error research to demonstrate how easy it is to make mistakes and explore error rates in creating spreadsheets and is considered a benchmark in spreadsheet error creation. Whilst the wall task may be considered ‘simple’ in the world of spreadsheet modelling and error, the task
may well be complex for a LLM and is unlikely to be included in the corpus of material
ChatGPT is trained on. This means that ChatGPT will be providing answers without any
existing information to fall back on and hence the output is going to be as extreme and as
“naked” as possible. Depending on the answers provided by ChatGPT, a series of other
prompts will be passed to clarify the details of the problem if needed. This in theory
makes solving the problem easier for ChatGPT.

1.4.2 Experiment 2: logic problems
Logical deduction problems starting at the easier end of the spectrum and increasing in
difficulty will be passed to ChatGPT. The easier puzzles are those that contain all the
information needed to come to right conclusion with minimal inference or deduction. The
medium difficulty puzzles are those that contain most of the information needed to come
to the right conclusion but require logical deduction and inference. The hard level puzzles
provide the bare minimum information needed and need significant logical deduction and
inference. The puzzles are sourced from Goodman (2023) and all puzzles are descriptive
word-based puzzles.

1.4.3 Experiment 3: Boolean Logic Problems
In this final stage of the experiment, Boolean logic questions will be posed to ChatGPT
that are drawn from a first year Bachelor of Science course in Computer Science in
Cardiff School of Technologies, Cardiff Metropolitan University. These questions probe
logical equivalences in Boolean logic achieved by combing of standard and negated logic
gates with negated input.

2.0 RESULTS
The follow section will discuss the results obtained from the multiple phases of the three
experiments conducted.

2.1 Experiment 1 Phase 1 Standard Wall Task
In this initial phase, ChatGPT is given the wall task without any modification. This task is
usually given to students to complete as a means to illustrate how even simple
spreadsheets can yield mistakes and that these mistakes can be difficult to notice and
correct.

If we give ChatGPT the unedited wall task, the result contains many mistakes. Before the
prompt was entered, ChatGPT was told it was going to act as an Excel programmer. The
prompt given was:

You are to build a spreadsheet model to help you create a bid to build the wall
around the garden.
You will offer two options, Welsh granite or Brick.
The wall will be built by a team of two people. The team will work three eight
hour days to build either type of wall.
The wall will be 20 feet long 6 feet tall and 2 feet thick.
Wages are £5 per hour per person. You will need to add 20% to the wages to
cover incentive bonuses.
Welsh granite costs £5 per cubic foot and brick costs £4 per cubic foot.
Your bid must add a 30% profit margin to your expected cost.
The response was:

\[ IF(A1="Welsh Granite", (20*6*2) *5 *2 *3 *1.2 *1.3, IF(A1="Brick", (20*6*2) *4 *2 *3 *1.2 *1.3)) \]

This output contains several problems and provides an incorrect result. The formulae will be broken down into its identifiable parts, examined for accuracy and how the placement of the text in the prompt influences the solution.

The first condition in the opening IF statement and the value if false which opens another IF statement, =IF (A1="Welsh Granite" and IF(A1="Brick", hinges on a text value in A1, either “Welsh Granite” or “Brick”.

The second part calculates the volume of the wall appearing as a calculation in brackets, (20*6*2). This calculation is correct and accurately reflects the instructions given but omits the material cost calculation which would appear to be a logical inclusion at this stage in the calculation.

The third part calculates the wage cost, some of the material costs and the profit margin, *5 *2 *3 *1.2*1.3. The first thing to note that unlike the volume calculation, the wage, material and profit margin costs are not in brackets. Secondly, what is omitted from this calculation is the 8 hours a day and £5 per hour wage parameters. The *5 relates to the Granite material costs (an additional query was passed to ChatGPT changing the £5 per hour to £6 per hour to test which “5” is being cited in this formula), the *2 relates to the number of workers needed, *3 relates to the three working days needed to complete the work and the *1.2 relates to the added 20% incentive bonus for the workers. The *1.3 relates to the 30% profit margin that should be added to the overall cost of the bid.

The lack of brackets mean that the calculation is completely inaccurate, there should be brackets around the whole statement (5 * 2 * 3) * 1.2)*1.3. Even this would not fix the problem though since the 8 hours a day and the £5 per hour aspects of the wage calculation are omitted from the answer, the logic of including the Granite cost per square foot in this part of the calculation is unclear. Incorporating the omitted elements would also mean that the calculation would have to be approached differently, a more logical approach would be to calculate the volume and material costs and then add those to the wage costs and then calculate the 30% profit margin.

2.2 Experiment 1 Phase 2

It is interesting that the solution offered by ChatGPT is an attempt to answer the whole problem in one cell. In order to see if the complexity of fitting everything in one cell is a limiting factor, another prompt was attempted adding the following sentence at the end of the problem description:

You should produce separate formulae for brick or granite costs, wage costs and profit margin calculations.

The thinking behind this is that perhaps a valid solution can be arrived at if the complexity of the formulae was reduced. ChatGPT output the following:

\[ Brick/Granite Costs: IF(A1="Welsh Granite", (20*6*2)*5, IF(A1="Brick", (20*6*2)*4)) \]
\[ Wage Costs: (20*6*2)*5*2*3*1.2 \]
\[ Profit Margin: (20*6*2)*5*2*3*1.2*1.3 \]

Here ChatGPT has produced some correct and incorrect calculations. The volume calculation is correct as it was in the previous example. The materials costs are provided
correctly, this is a change to the previous solution offered, perhaps because the prompt provided to ChatGPT explicitly asked for the brick and granite costs. However, both the wage costs and the profit margin are incorrect. The wage cost still omits the 8 hour day calculation and both the wage and profit margin calculations do not use any brackets, other than the volume, to ensure the order of calculation is correct.

2.3 Experiment 1 Phase 3

A modified prompt was passed to ChatGPT which contained the problem statement as already stated but included the following addition.

> The wages should be calculated on the number of hours worked in three days at £5 per hour with a 20% bonus added to the result.

It was hoped that by explicitly stating the correct method for calculating wages that ChatGPT’s response would be more accurate. ChatGPT produced the following:

Material Costs: \( IF(A1=\text{"Welsh Granite"}, (20*6*2)*5, \)
\( IF(A1=\text{"Brick"}, (20*6*2)*4) \)
Wage Costs: \( (2*3*8)*5*1.2 \)
Profit Margin: \( (20*6*2)*5*2*3*8*5*1.2*1.3 \)

The solution provided shows a correct wage cost calculation with no omissions and correct use of brackets. However, the profit margin calculation, which brings together the wage, material, bonus and profit margin costs does not use brackets to ensure the calculation is executed in the proper order. The wage cost in the profit margin calculation is different to the one provided under wage costs, the difference being missing brackets.

ChatGPT was then asked to provide the solution as one formula, the response was as follows:

\[
=IF(A1=\text{"Welsh Granite"}, \((20*6*2) *5 *2 *3 *8 *5 *1.2 *1.3\), IF(A1=\text{"Brick"}, \((20*6*2) *4 *2 *3 *8 *5 *1.2 *1.3)))
\]

Clearly from the above outcome, the original omission made by ChatGPT is corrected but the calculation is not correct because of the incorrect order of computation through a lack of brackets. The correct method using brackets for calculating the wage cost is abandoned in favour of a solution that closely resembles the original solution provided by ChatGPT.

2.4 Experiment 2 Phase 1

In this initial phase of the experiment, easy logic problems are posed to ChatGPT. These easy problems contain all of the information needed in the question and require no or little logical deduction or inference.

The prompt posed was:

> Six children (Leisha, Benito, Delia, Charlotte, Weldon, and Zina) were measured yesterday. Weldon is taller than Delia but shorter than Zina. Leisha is taller than Benito but shorter than Delia and Weldon. Benito is not the shortest. Can you put them in order of height from tallest to shortest?

The answer provided by ChatGPT was:

Zina, Weldon, Delia, Leisha, Benito, Charlotte

Zina is the tallest, as Weldon is taller than Delia but shorter than Zina. Weldon is next, followed by Delia. Leisha is taller than Benito but shorter than Delia and
Weldon, so she is next. Benito is not the shortest, so he must be next. Charlotte is the shortest, as she is the only one left. Therefore, the order from tallest to shortest is Zina, Weldon, Delia, Leisha, Benito, Charlotte.

As can be seen from the answer, ChatGPT has correctly solved the problem and cited the correct deduction from the text. This success is likely due to the minimal deduction needed and the completeness of the information in the prompt.

2.5 Experiment 2 Phase 2

In this phase of the experiment, more complex logic puzzles were presented to ChatGPT and the outcome observed.

The first prompt given to ChatGPT:

Each of five neighbourhood dogs (Saber, Ginger, Nutmeg, Pepper, and Bear) is enjoying one of the following activities: getting its ears scratched, playing catch, taking a nap, burying a chew toy, and going for a walk.

Pepper is either playing catch or burying a chew toy. Neither Ginger nor Saber nor Bear is on a walk. One of the dogs named after a spice is getting its ears scratched. A dog not named for a spice is playing catch. Bear is getting some exercise. Can you figure out what each dog is doing?

The answer and reasoning provided by ChatGPT to the prompt:

Saber: Taking a nap ginger: Burying a chew toy Nutmeg: Getting its ears scratched Pepper: Playing catch Bear: Going for a walk

This is based on the clues given in the question. Pepper is either playing catch or burying a chew toy, so one of those two activities must be what it is doing. Neither Ginger, Saber, nor Bear is on a walk, so none of them are doing that activity. One of the dogs named after a spice is getting its ears scratched, so that must be Nutmeg. A dog not named for a spice is playing catch, so that must be Pepper. Finally, Bear is getting some exercise, so it must be going for a walk.

The answer provided is inaccurate and the reasoning provided contains several contradictions. The correct answer to the problem is as follows: Saber is asleep, Ginger is getting her ears scratched, Nutmeg is on a walk, Pepper is burying a chew toy and Bear is playing catch.

This conclusion is reached through a series of deductions, the first two clues reveal which dog is on a walk, since either Pepper is playing catch or burying a chew toy AND neither Ginger, Saber or Bear is on a walk, that only leaves Nutmeg left.

The second step is given in the third clue, one of the dogs named after a spice is getting its ear scratched. Since we know Nutmeg is on a walk and Pepper is either burying a chew toy or playing catch, it can only be Ginger since neither Bear nor Saber are named after a spice.

The third step is given in the fourth, fifth and sixth clues, a dog not named after a spice, either Bear or Saber, is playing catch. The fourth clue reveals it must be Bear since Bear is getting some exercise but according to the sixth clue is “not on a walk”. This means Bear must be playing catch since that is the only activity left that counts as exercise. This also means that Pepper must be burying a chew toy since she is not playing catch and that Saber must be asleep since this is the only activity not accounted for at this stage.
The answer provided by ChatGPT is mostly wrong and the reasoning cited is contradictory in places. The only correct assertion in the answer is that Saber is asleep. All of the other conclusions are incorrect and there is little evidence of logical reasoning. A few examples of this will now be highlighted:

When deducing which dog is getting its ear scratched, ChatGPT states the following:

*One of the dogs named after a spice is getting its ears scratched, so that must be Nutmeg*

At this stage, there are several candidates who could fit the description in the problem but Nutmeg is not one of them. Nutmeg is identified in the first clue as going on a walk since it cannot be Pepper and through negation it cannot be Saber, Bear or Ginger.

When deducing what activity Pepper is involved in, ChatGPT states:

*A dog not named for a spice is playing catch, so that must be Pepper*

Here ChatGPT doesn’t seem to recognise that Pepper is named after a spice although it could possibly think that Pepper is a vegetable, in either case the answer provided is incorrect and Pepper is a commonly known spice. It is unclear whether the *not* is a recognised logical operator here since the response ignores the implication of *not*.

When deducing what activity Bear is involved in, ChatGPT states the following:

*Finally, Bear is getting some exercise, so it must be going for a walk.*

However, this contradicts the second clue that states:

*Neither Ginger nor Saber nor Bear is on a walk*

So it is unclear why ChatGPT would come to the conclusion that Bear is on a walk since it is explicitly stated otherwise. It would appear that *Neither* and *nor* are not understood logical operators, this may explain why ChatGPT came to the incorrect conclusion about Bear being on a walk.

### 2.6 Experiment 2 Phase 3

The final prompt passed to ChatGPT is as follows:

*The Reds, the Grays, the Blues, and the Greens have a round-robin tournament. Each team plays each other team once, for a total of six games. The Greens won more games than the Blues. The Grays lost more games than the Blues. The Reds tied the Greens. (This was the only tie in the tournament.) Who won the game between the Reds and the Blues?*

The correct response and reasoning is as follows: The Reds won the game between the Reds and the Blues. The Greens tied one game, so they won a maximum of two games. The Greens won more than the Blues, so the Blues won a maximum of one game. Since the Greys lost more times than the Blues, the Greys must have lost all three matches. The match that the Blues won must have been against the Greys, so the Blues lost the match against the Reds.

The response from ChatGPT is as follows:

*The Reds and the Blues did not play each other, so there is no winner for that game.*
The Reds and the Blues did not play each other in the round-robin tournament, so there is no winner for that game. The clues given in the question only provide information about the results of the other five games. The Greens won more games than the Blues, the Grays lost more games than the Blues, and the Reds tied the Greens. Therefore, there is no winner for the game between the Reds and the Blues.

The response from ChatGPT is a little surprising but incorrect. The conclusion that the Reds and Blues did not play each other is contradicted in the prompt by two statements, the first is the definition of a ‘Round Robin’ and the second is that there were six games played.

The Reds, the Grays, the Blues, and the Greens have a round-robin tournament. Each team plays each other team once, for a total of six games.

This total failure to compute the answer is no doubt based on the lack of information available to ChatGPT in the prompt. For instance, in the response to the prompt, ChatGPT complains that there is no information about the Red versus Blue game, The clues given in the question only provide information about the results of the other five games and therefore incorrectly concludes that there was no game between Red and Blue.

2.7 Experiment 3 Phase 1

In this phase, ChatGPT was asked to provide all of the standard and negated logic gate truth tables. The response obtained from ChatGPT was a complete list of correct truth tables covering OR, AND, XOR, NOT, NOR, NAND and XNOR.

ChatGPT was also asked what negated input in truth tables was, ChatGPT provided the following correct answer:

Negated input in truth tables is when the input is reversed. For example, if the input is A and B, then the negated input would be NOT A and NOT B. This means that the output of the truth table will be the opposite of what it would be with the original input.

This first phase was completed to establish if ChatGPT could demonstrate logic gates, truth tables and negated input needed in phases 2 and 3.

2.8 Experiment 3 Phase 2

Given the responses from ChatGPT in experiment 3 phase 1, a correct explanation of negated input and the correct reproduction of logic gate truth tables, the following prompt was posed:

Does a negated input NOT gate produces an output that is the same as the original input?

ChatGPT responded:

No, a negated input NOT gate produces an output that is the opposite of the original input.

This answer is incorrect, the correct response is yes, a negated input not gate produces an output that is the same as the input:

<table>
<thead>
<tr>
<th>A</th>
<th>NOT</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
</tr>
</tbody>
</table>
2.9 Experiment 3 Phase 3

Given the responses from ChatGPT in experiment 3 phase 1, a correct explanation of negated input and the correct reproduction of logic gate truth tables, the following prompt was posed:

*Does a negated 2-input OR gate produce an output that is the same as a 2-input NAND gate?*

The correct answer to this question is yes, these two combinations of OR, AND and NOT are equivalents:

<table>
<thead>
<tr>
<th>-A</th>
<th>-B</th>
<th>OR</th>
<th>A</th>
<th>B</th>
<th>NAND</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>1</td>
</tr>
</tbody>
</table>

ChatGPT provided the following response:

*No, a negated 2-input OR gate does not produce an output that is the same as a 2-input NAND gate. A 2-input OR gate produces an output that is the logical sum of its two inputs (i.e., if either one or both of the inputs is high, the output will be high). On the other hand, a 2-input NAND gate produces an output that is the logical product of its two inputs (i.e., both inputs must be high in order for the output to be high). Therefore, the output of a negated 2-input OR gate is not the same as the output of a 2-input NAND gate.*

The response is incorrect, the reasoning cited is partially correct and partially incorrect. The first assertion made by ChatGPT is correct but subsequent assertions are incorrect, the responses seem to ignore the implication of NOT.

The definition of OR is correct: A 2-input OR gate produces an output that is the logical sum of its two inputs (i.e., if either one or both of the inputs is high, the output will be high). The error comes when discussing NAND, a 2-input NAND gate produces an output that is the logical product of its two inputs (i.e., both inputs must be high in order for the output to be high). Here ChatGPT has given the definition of an AND gate and has ignored the NOT implication.

3.0 CONCLUSIONS

The experiments have provided some interesting insight into how capable ChatGPT is in different scenarios. This section will now address the research questions and hypotheses identified at the start of the paper.

3.1 Research Question and Hypothesis 1

Research question 1 and hypotheses 1 were tested in experiment 1, phases 1, 2 and 3 and in experiment 2, phases 1, 2 and 3. In the first set of experiments, the wall problem was posed to ChatGPT.

In experiment 1 phase 1, ChatGPT made some significant errors in the solution it offered. It made errors of omission when calculating the wage costs and it made BODMAS errors in the final calculation which meant the answer was completely wrong. The conclusion
reached was that the problem description was too complex, phases 2 and 3 attempted to reduce the uncertainty and problem solving needed.

In experiment 1 phase 2, the prompt asked for formulae for all of the sub tasks to be expressed separately, rather than in one long formula which ChatGPT naturally defaults to. This resulted in some improvement, the material costs were correctly calculated but the wage cost still omitted relevant information and BODMAS was still an issue.

In experiment 1 phase 3 explicit instructions on exactly how to calculate wage costs were included in the prompt passed to ChatGPT. The results obtained showed some improvement, the wage cost was correctly calculated with the correct use of BODMAS as a separate calculation. However, when ChatGPT was asked to use the solution in one formulae, it kept the correct method for calculating wages but abandoned the correct use of BODMAS making the result totally inaccurate.

The BODMAS errors in all three phases of experiment 1 are consistent with other observations on the mathematical failures of ChatGPT although BODMAS is not explicitly addressed (Frieder, et al., 2023; Borji, 2023).

In the second series of experiments ChatGPT was passed a series of logic puzzles, some of which contained all of the information needed to compute the correct answer, some of which needed deduction and inference to solve them correctly.

In experiment 2 phase 1, a simple sorting logic puzzle with no uncertainty or inference was posed to ChatGPT. The result provided was completely correct.

In experiment 2 phase 2, a relatively more complex logic puzzle than phase 1 was used which required some inference. The response generated was incorrect and contained contradictions and errors which seem to stem from the use of negation in the description of the problem. The response contained only one correct assertion and ChatGPT would seems to ignore the use of negation, specifically the use of the terms **neither** and **nor**.

In experiment 2 phase 3, a complex logic puzzle was posed to ChatGPT that needed significant amounts of inference and deduction to compute the right answer. ChatGPT gave a slightly unexpected answer to the question, the answer provided contested the assertion made in the prompt given to it on the basis that no information was provided about the specific prompt posed to ChatGPT.

In terms of answer research question 1, the evidence generated from the experiments suggests that ChatGPT has very limited ability to generate valid code where it is required to problem solve the question. This is mostly evident in experiment 1 where the initial answer provided to the unedited version of the wall task contained some correct but mostly incorrect responses to the prompt posed. In phases 2 and 3, the prompt was simplified in an effort to reduce the problem solving needed and make the requirements for the computation more explicit. This simplification did improve the accuracy of the responses but it did not result in a completely correct solution.

This is also supported by the outcome of experiment 2, where logic deduction puzzles were posed to ChatGPT which contained varying levels of uncertainty. Uncertainty, deduction and inference all negatively impact the ability of ChatGPT to respond correctly as shown in the results to phases 2 and 3 of experiment 2. According to Bang et al. (2023), ChatGPT shows limited ability in inductive reasoning and in mathematical reasoning in general, a finding also echoed other researchers (Davis, 2023; Frieder, et al., 2023)

Hypothesis 1 is therefore upheld, the amount of uncertainty in the prompt does limit the accuracy of the solutions provided, in the experiments conducted, uncertainty comes in
the form of a lack of complete information and the amount of deduction and inference needed to compute the right answer. In scenarios where there is no deduction or inference and complete information about the purpose of the task, the performance of ChatGPT is efficient and accurate.

3.2 Research Question and Hypothesis 2

Research question and hypothesis 2 was designed to probe why ChatGPT struggles with logical deduction, inference and uncertainty. The research question and hypothesis are answered by experiments 2 and 3.

Experiment 2 showed that as the prompt demanded more uncertainty, logical deduction and inference, the accuracy and performance of ChatGPT diminished.

Other authors have experimented with logical reasoning and ChatGPT, posing similar logic puzzles to those contained in this paper and noting the outcome (Borji, 2023). The results broadly agree with the outcome of the experiments in this paper, ChatGPT is unable to deduce the correct answers to these puzzles, no real reasoning is given for this however.

The results from experiment 3 show that despite ChatGPT having all of the information needed to compute the queries put to it, it was unable to come up with a correct assertion. This is evidenced by the correct definition of negated input and production of logic gates and truth tables for common logic operators and not combinations such as AND, OR, XOR, NAND, NOR, XNOR and NOT. However, when logic equivalence questions were posed to it, it was unable to arrive at the right answer and provided incorrect reasoning. Although many parts of the reasoning were incorrect, ChatGPT seems to have a particular issue with negation. Some of the reasoning provided would have been true if the prompt didn’t included gates that are negated such as NAND or NOR.

The answer to research question 2 is that ChatGPT has only shallow “knowledge” of logic operators and although it can correctly cite the function and composition of logic gates, it is unable to apply this knowledge in the queries posed to it. Further, ChatGPT doesn’t really “know” anything, whilst it may be able to provide the correct information given a prompt passed to it, ChatGPT doesn’t “understand” the information it is passing to the user, it is simply the most probable answer given the input prompt. Indeed, all of ChatGPTs “knowledge” is the same, the answers provided are merely labels that have a high probability of fulfilling the intent of the prompt.

Some studies (Wang, et al., 2023) probe the use of Boolean logic for literature searching and achieve comparable results to other tools, however they do not specifically test ChatGPTs ability to reason with Boolean logic. So whilst there a consensus that ChatGPT struggles with written logic puzzles, this paper is novel in presenting results obtained from Boolean logic expressions hence there is no secondary work to compare these results with.

Hypothesis 2 is upheld, ChatGPT has only shallow, surface knowledge of logical reasoning and as such it’s ability to provide logic deduction is almost entirely based on the availability of information in the prompt, if some information is missing, ChatGPT is unable to fall back on logical reasoning to solve the problem and instead produces incorrect solutions where it is not explicitly told how to reason (Bang, et al., 2023; Borji, 2023; Frieder, et al., 2023).
3.3 Other observations on ChatGPT

During the development and execution of the experiments, a number of interesting observations were made about the behaviour of ChatGPT. This section will explore some of these observations and what can be learnt from it.

3.3.1 Inconsistent use of BODMAS

When executing the wall task experiment, without exception, ChatGPT would always use the correct formulae for calculating volume including the correct use of BODMAS. This is in contrast to other solutions provided which contained incorrect use of BODMAS, resulting in incorrect calculations. At first this was puzzling but it was realised that it was consistently able to provide a correct calculation for volume because it is highly likely that the formulae for volume is explicitly in the large language corpus on which ChatGPT is trained. So it’s able to cite and calculate volume correctly because it has already learnt the explicit way in which it should be calculated. However, when it is required to apply BODMAS in a situation not explicitly covered in the training of the model, it is unable to use the same principles correctly, inconsistencies in logic and mathematical reasoning are cited by many studies (Borji, 2023; Bang, et al., 2023; Davis, 2023; Frieder, et al., 2023).

The same may also be said of the correct answers provided around negated input and various logic gates. It is highly likely that these gates form part of the large language corpus and hence it can reproduce them accurately. However, when it faces novel situations where it has to apply that knowledge, it is unable to complete this task since it is not explicitly covered in its training phase. If the exact questions used in these experiments had been part of the training corpus, then it’s likely that ChatGPT would have provided the correct answers.

3.3.2 False statements

The manner in which the prompt is worded makes a great difference to how ChatGPT responds. When posing logic questions in experiment 3, the questions were lifted from a class on Boolean logic. The unedited wording of the questions started “Show that” a negated two input OR gate produces the same output as 2 input NAND gate”. Specifically, using “show that” phrase induced ChatGPT into inventing false proofs and reasoning to “show that” the two outputs are the same. This was confusing when initially encountered because the correct answer to the question is yes, they are equivalents but in examining the reasoning behind the answer generated, it becomes clear that the reasoning provided is false.

To test this idea further, a prompt was posed that said “Show that a 2 input AND gate produces the same output as a 2 input OR gate”. This is obviously impossible since the outputs of these two gates are different. The reasoning in response to this impossible question is false and contradicts the accurate truth table provided by ChatGPT:

<table>
<thead>
<tr>
<th>A</th>
<th>B</th>
<th>A AND B</th>
<th>A OR B</th>
</tr>
</thead>
<tbody>
<tr>
<td>T</td>
<td>T</td>
<td>T</td>
<td>T</td>
</tr>
<tr>
<td>T</td>
<td>F</td>
<td>F</td>
<td>T</td>
</tr>
<tr>
<td>F</td>
<td>T</td>
<td>F</td>
<td>T</td>
</tr>
<tr>
<td>F</td>
<td>F</td>
<td>F</td>
<td>F</td>
</tr>
</tbody>
</table>

In this truth table, the output of the 2-input AND gate is the same as the output of the 2-input OR gate. This is because when either A or B is true, the output of the AND gate is false and the output of the OR gate is true. When both A and B are
false, the output of the AND gate is false and the output of the OR gate is false. This shows that a 2-input AND produces an output that is the same as a 2-input OR gate.

This issue is overcome by altering the wording of the prompt, instead of using “Show that”, the prompt was rephrased as an explicit question using “Does a” as the opening phrase. That change flips ChatGPT to evaluate the truth of the statement which it does correctly:

Does a 2 input AND gate produce an output that is the same as a 2 input OR Gate?

No, a 2-input AND gate produces an output that is the opposite of a 2-input OR gate. The output of a 2-input AND gate will be high (1) only if both inputs are high (1). The output of a 2-input OR gate will be high (1) if either or both inputs are high (1).

The answer provided by ChatGPT is simply trying to fulfil the request posed to it, hence the answer does “show that” AND = OR even if that is completely incorrect. ChatGPT suffers from “Hallucinations” as do all LLMs (Bang, et al., 2023; Davis, 2023). These hallucinations can cause ChatGPT to make difficult to verify statements that seem plausible but do not stand up to scrutiny and are arbitrary in nature. The AND = OR examples would seem to be a combination of a hallucination and an eagerness to please the user.

3.4 Limitations

This work merely scratches the surface of what the limitations and strengths of ChatGPT for spreadsheet code generation. A much larger and more exhaustive study is required to confirm the conclusions of this work and to understand the scope in which ChatGPT can be accurate and where its hard limits in ability exist.

One could argue that giving ChatGPT queries that require deduction and inference with incomplete information is not a fair test of the technology and that no claims were made around such abilities by the vendors of the software. Whilst this is true, it is likely that humans using this technology will do so in an imperfect way and it may become a tool for problem solving issues or for bridging gaps in programming knowledge, hence ChatGPT and other LLM presents anew kind of spreadsheet user risk

Whilst the data in this paper has revealed some interesting subtleties around how prompts are worded, much more study is needed to fully understand these issues. The emerging discipline of “Prompt Engineering” will be an important methodology for leveraging the most value from ChatGPT (White, et al., 2023; Short & Short, 2023).

3.5 Future work

From these experiments, some areas of future work are identified:

The first would be a more exhaustive examination of the programming pitfalls and strengths of ChatGPT. This could take the form of further experiments into producing spreadsheet formulae with uncertainty in the prompts to find the limits of what is possible with the technology. The use of language in the prompt seems to be particularly significant to the responses generated by the software, some detailed examination of how ChatGPT responds to the use of different language phrasing could be beneficial to understanding how to best exploit the technology. This could establish the foundation of “Prompt Engineering” for spreadsheet applications based on some of the findings for general purpose programming (White, et al., 2023; Short & Short, 2023)
The second area of future work would be an empirical study of spreadsheet modellers who use ChatGPT to produce formulae. This study could examine the way in which the technology is being used by real spreadsheet modelers including their approach to generating, testing and verifying the output of ChatGPT. This goal of this work would be to understand best practice in spreadsheet modelling with ChatGPT and to understand the practical strengths and pitfalls of the technology.

The third area of future research could experimentally test the efficiency and efficacy of spreadsheet modellers using ChatGPT versus spreadsheet modellers not using ChatGPT. The aim of this work would be to understand the usefulness of the approach across different levels of experience and could be measured in terms of how accurate or otherwise each group is at generating spreadsheets for certain problems and how much time and effort is invested into producing the answer.

The final area of future work would be compare different AI based approaches to creating spreadsheets. For instance, contrasting LLMs with alternative approaches that leverage AI in different ways such as (Hofer, et al., 2017; Thorne, et al., 2013)

3.6 Conclusion

ChatGPT is a powerful resource that has great utility in certain activities, particularly for natural language generation. It also has some ability in code generation. Where the prompt is offered in complete detail, in these circumstances ChatGPT provided consistently correct code. However, as these experiments have show, if there is any uncertainty, inference or deduction needed from the prompt, ChatGPT has questionable ability to provide accurate code or reasoning. This opens a new front in spreadsheet risks, those that arise from the use of LLM generated spreadsheet formulae.
REFERENCES

Bang, Y. et al., 2023. A Multitask, Multilingual, Multimodal Evaluation of ChatGPT. Pre-print Arxiv, Centre for Artificial Intelligence Research (CAiRE) The Hong Kong University of Science and Technology, 2302.04023.pdf (arxiv.org)


Davis, E., 2023. Mathematics, word problems, common sense, and artificial intelligence, Pre-print Arxiv, Department of Computer Science, University of New York, 2301.09723.pdf (arxiv.org)


White, J. et al., 2023. A Prompt Pattern Catalog to Enhance Prompt Engineering with ChatGPT, Pre-print Arxiv, Department of Computer Science Vanderbilt University, 2302.11382.pdf (arxiv.org)

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 image courtesy of EuSpRIG Authors.