End-user encounters with lambda abstraction in spreadsheets: Apollo's bow or Achilles' heel?

Advait Sarkar[1,2], Sruti Srinivasa Ragavan[1], Jack Williams[1], Andrew D. Gordon[1,3]
[1]*Microsoft Research, Cambridge, United Kingdom*
[2]*University of Cambridge, United Kingdom*
[3]*University of Edinburgh, United Kingdom*
{advait, a-srutis, jack.williams, adg}@microsoft.com

***Abstract*—The value of computational abstractions to non-expert end-user programmers is contentious. We study reactions to the LAMBDA function in Microsoft Excel, which enables users to define their own functions using the spreadsheet formula language, through a thematic analysis of nearly 2,700 comments posted on the Reddit, Hacker News, YouTube, and Microsoft Tech Community online forums. We find that computational abstractions are viewed both as helpful and harmful, that users encounter learning and understanding barriers to applying them, and that there are deficiencies and opportunities in tooling such as in formula editing, versioning, reuse and sharing. We find that the introduction of LAMBDA prompts new debate around whether spreadsheets are code, whether writing formulas can be considered programming, and whether spreadsheet users identify themselves as programmers.**

***Index Terms*—human-computer interaction, content analysis, end-user software engineering, functional programming**

I. Introduction

In December 2020, the LAMBDA function was made available to Excel spreadsheet users, enabling them for the first time to create computational abstractions in spreadsheets, by defining new formula functions using the language of spreadsheet formulas alone [1–3].

This development is of interest to end-user programming research because, despite the status of the spreadsheet as the most widely-used programming environment, no commercially dominant spreadsheet package had implemented a language-native method for defining new functions; prior solutions relied on parallel scripting languages. Thus, the spreadsheet formula language remained an anomaly: the only widely-used programming language that did not support subroutine definition since the invention of subroutines in the 1940s [4, 5].

The limitations of the lack of a native computational abstraction in spreadsheets are well-documented (Section II). In response, researchers have explored various abstraction mechanisms. The Forms in Forms/3 spreadsheets can be invoked as subroutines [6], and the design of sheet-defined functions [7, 8] demonstrates the integration of such abstractions into conventional spreadsheets. Nevertheless, such abstraction techniques were not implemented in commercial spreadsheets.

To deal with these limitations, users (with sufficient expertise) either resorted to using abstraction mechanisms offered using alternative programming paradigms (such as VBA macros in Excel, or Javascript custom functions in Google Sheets), or installing add-ins which introduced new language features, or alternating between spreadsheets and other programming languages [9]. Alternatively, some users developed workaround craft practices, such as the creation of templates for sections of grid that could be copied and pasted as self-contained computational abstractions [10], or built 'megaformulas': elaborate formulas constructed as individual parts but eventually 'rolled up' (nested) into a single cell for reuse.

Despite such evidence of abstractions in real-world spreadsheets, and the promise of lambda abstractions, it is unclear *prima facie* whether the LAMBDA construct is a net gain for end-user programmers. For example, abstraction can make programming more powerful, and programs more concise and intelligible; on the other hand, they can make programs hard to understand, and poorly-considered application of abstractions can cause problems for downstream users of programs [11].

The introduction of such an abstraction capability to spreadsheets therefore raises two key questions:

- What are the implications of computational abstractions for the end-user experience of spreadsheets?
- What barriers do end-user programmers face when learning and using abstractions, and what strategies do they employ to overcome them?

To investigate these questions, we analysed discussions from online communities about the LAMBDA function in Microsoft Excel (Section III). While our analysis does not yet answer these questions in depth, our findings identify a broad set of phenomena, helping to map and scope further study.

We find several implications for the user experience of spreadsheets, such as the potential for abstraction to help as well as hinder comprehension, and the inadequacy of formula authoring and management tools; we find that users encounter several barriers to applying these abstractions and employ several strategies to address them; and we find that computational abstractions play a crucial role in the formation of user identities as programmers (Section IV). Thus, the lambda abstraction in spreadsheets can either be a powerful tool, as the golden bow of Apollo, or it can be an Achilles' heel, weakening comprehension and community participation.

978-1-6654-4214-5/22/$31.00 ©2022 IEEE

II. Background

A. The Excel LAMBDA function

Lambda abstraction is a concept from the lambda calculus, introduced by Church as a foundation for mathematics [12, 13], and long considered a foundation for programming language concepts [14, 15]. Its implementation in Excel is via a formula function with the same name. To distinguish between the concept of lambda abstraction (as well as specific instances of function values, also known as lambdas), and the Excel function, we style the Excel function as LAMBDA.

The LAMBDA function takes as arguments a list of parameters followed by a formula that may refer to those parameters. For example, the following call creates a lambda that adds two numbers: `=LAMBDA(x,y,x+y)`.

Placing such a call to LAMBDA in a cell produces an error, because in the Excel implementation, a function value cannot be the result of a cell. Instead, the lambda can be directly invoked following its creation: one can write `=LAMBDA(x,y,x+y)(2,3)` in a cell to yield the result 5.

To introduce a new user-defined function using a LAMBDA formula, a name can be assigned to the formula using the name manager.[1] It can then be invoked via the name. For example, one can assign the name `AddNumbers` to `=LAMBDA(x,y,x+y)` and invoke it, within a cell, as `=AddNumbers(2,3)` to yield the result 5.

B. Abstraction in end-user programming

In the Cognitive Dimensions of Notations framework, Green and Petre define programming abstraction as the *"grouping of elements to be treated as one entity"* [16]. *Data* abstractions, such as arrays or tables, allow us to perceive and operate on a group of data as a single entity. *Control* or *computational* abstractions include conditionals, loops, and subroutines. LAMBDA functions fall into the latter category.

In Cognitive Dimensions terms, a notation can be abstraction-hating, abstraction-tolerant and abstraction-hungry, based on whether it prohibits, permits, or mandates the creation of new abstractions by the programmer. LAMBDA takes the spreadsheet formula language from being control abstraction-*hating* to a control abstraction-*tolerant* language.

Control abstractions are difficult for end-user programmers to reason about [17]. In notating multiple possible paths of execution, they resist direct manipulation interfaces [18]. Moreover, end-user programmers avoid developing control abstractions due to the attention investment required [19] and challenges when learning and creating them [16]. Programming by demonstration [20] and programming by example [21] aim to address these difficulties by bridging direct manipulation and control abstractions through inference.

C. Proposed control abstractions in spreadsheets

Sheet-defined functions (SDFs) enable users to define their own computational abstraction using the spreadsheet formula language, much like LAMBDA [7, 22]. However, SDFs differ from LAMBDA in that the function body can be defined from a collection of cells, rather than a single formula.

A user study of SDFs found that while having a reusable abstraction was helpful, in many cases participants preferred to view the body of the function when invoked [8]. This contrasts with the *hiding* of the body and intermediate values that is inherent to the user experience of subroutines. In response, Gridlets, a non-hiding computational abstraction, has been proposed [10, 23] but not evaluated in an empirical study.

Another approach is to adopt practices from software development to spreadsheet development, for better long-term maintainability of spreadsheets. This is end-user software engineering [24], end-user development [25], or spreadsheet engineering [26]. ClassSheets [27] is an example of an engineering abstraction developed to manage spreadsheet complexity. End-user development research has also considered other engineering practices such as testing [28].

D. Spreadsheet abstraction in practice

Complex models continue to be built using spreadsheets despite the lack of a native computational abstraction, resulting in challenges for correctness, maintenance and auditing [29]. In response, spreadsheet users have adopted various alternatives.

Expert users build abstractions using extensibility capabilities via a different language; examples are VBA macros in Microsoft Excel and JavaScript extensions in Google Sheets. Less expert users use third-party add-ins (e.g., Ablebits[2], ASAP Utilities[3]) that provide utilities for common tasks to augment the built-in function library. Others delegate the development of complex spreadsheets to others with more (or different) expertise [30].

Users have also developed practices to manage the complexity of the spreadsheet logic, to abstract away details and to ease comprehension and maintenance. For example, the FAST standard[4] for spreadsheet financial modeling recommends that formulas take no more than 24 seconds to explain, and to move intermediate calculations to separate sheets.

However, the adoption and effectiveness of such standards are limited. There is lack of awareness, and most spreadsheet users lack formal training. Moreover, users are not motivated to invest attention in learning and adopting these practices. End-user programmers are typically task-oriented, and good design is not a necessity for task completion [28].

III. Method

A. Approach

A feature like LAMBDA can be studied in many ways. An experiment with authoring or comprehension tasks can evince learning barriers or usability issues, while interviews with experienced users (if such a group could be recruited) can yield insights about LAMBDA use and practices in users' workflows.

Our motivation in this study is to observe the interaction of a broad range of end-user programmers with LAMBDA.

[1] https://aka.ms/ExcelNameManager

[2] https://www.ablebits.com/

[3] https://www.asap-utilities.com/

[4] https://www.fast-standard.org/

We were also interested in how perceptions of a new feature like LAMBDA can develop and evolve in discussion amongst end-user programmers, something that is difficult to study in interview or experimental settings.

We therefore opted to study discussions of LAMBDA in online communities. Communities such as Hacker News, Twitter and GitHub form a social ecosystem in which programmer relationships are formed and enacted [31] and studying them *"can yield insights into qualitative research topics, with results comparable to and sometimes surpassing traditional qualitative research techniques"* [32]. Barik et al. applied this method to investigate notions of play in programming [33], and we draw upon the method of that study.

To our knowledge, ours is the first systematic qualitative study of online communities in spreadsheet research.

B. Dataset

One researcher compiled a list of potential sources, consisting of discussion threads matching the query "Excel Lambda" using the search functions built into Hacker News, Reddit, Microsoft Tech Communities and YouTube as of 20 January 2022. From the first three, we extracted discussion threads related to "Excel Lambda"; for YouTube, the search returns matches based on video titles and descriptions; the comments associated with these videos were taken to be the corresponding "threads". Additional forum sources were inspected but not included because they contained too few (fewer than 100) relevant comments. After removing false positives in the threads by manual inspection, we gathered comments from the remaining threads using automated scripts.

Our final dataset, summarised in Table I, contained 361 distinct threads containing a total of 6999 comments authored by 3422 distinct user IDs. In total, this corpus contains 254,541 words of discourse either directly related to LAMBDA or occurring in a thread about LAMBDA, with an average of 36.37 words per comment. The data spanned the time period from 3 April 2008 through 1 February 2022. Even though a small proportion of these comments (6.9%) were written before the initial public release of LAMBDA on December 3, 2020, we retained them in case any interesting observations had been made in users' speculations about lambdas in spreadsheets.

C. Analysis

We used general thematic analysis with iterative coding and codebook development [34].

Initially, three researchers independently open-coded the same set of 400 comments (100 comments randomly sampled from each of the four sources). Together the researchers generated 94 proto-codes, which after discussion and negotiations led to an initial codebook of 29 codes. The three researchers then independently re-coded the same sample with the codebook. Manual inspection showed poor inter-rater agreement, consistent with initial open coding rounds in prior studies [35].

The three researchers then discussed coding disagreements and ambiguities and revised the codebook. They used the updated codebook to independently code a fresh random sample of 200 comments (50 from each source). The average inter-rater agreement in this round was 0.60 (Jaccard index). Disagreements were negotiated, and the codebook revised.

The final codebook, presented in Appendix A, consisted of 28 codes; 14 code definitions were updated and 2 were merged from the initial codebook following three rounds of negotiations between researchers.

Non-English comments could not be reliably interpreted by our English-speaking research team, and several comments were not relevant to Lambdas (e.g., "great video"). We coded these as irrelevant. These accounted for a sizeable fraction of the data (YouTube: 62%, Reddit: 54%, Hacker News: 48%, Microsoft Tech Community: 66%). Thus the findings presented in Section IV draw from a pool of 2697 comments.

With this final codebook, the dataset was divided into four sets and one of four researchers (three being the coders from previous rounds, the fourth an observer during previous negotiations) coded each set. As standard validation practice [36, 37], a sample of at least 10% of code assignments made by each researcher was audited and re-coded by another; this had an acceptably high agreement of 0.86.

Finally, the researchers grouped codes into larger themes, discussed overall findings and selected representative quotes.

Our use of inter-rater agreement is primarily as a negotiation aid, since our findings do not hinge on frequency counts. In our analysis, codes are the process, not the product; our findings are organised into themes which were synthesized from these codes through the negotiation of expert researchers. In doing so, we align with McDonald et al.'s state-of-the-art analysis and guidelines for reliability in CSCW and HCI research [36]. To connect our findings to our codebook, direct references to codes are presented **in bold**.

IV. Results

LAMBDA has several implications for the user experience of spreadsheets. It introduces a tension between abstraction and comprehension, and commenters propose an array of craft practices to cope. It also exposes limitations and opportunities in tooling. Commenters encounter barriers in learning and applying LAMBDA, and employ a variety of strategies to deal with these barriers. Moreover, LAMBDA leads commenters to critically evaluate their identity as programmers. Commenters contrast LAMBDA with alternatives to situate it within the landscape of available tools. Finally, we observed differences in topics and concerns between the various communities we studied. These findings are now detailed in turn.

A. Implications for the user experience of spreadsheets

1) The tension of abstraction: Abstraction is a double-edged sword and commenters seized upon this tension. While the benefits of LAMBDA are described in terms of improved comprehension, debugging, and maintenance, in each of these areas commenters spotted the potential both for improvement and regression.

For example, commenters noted how LAMBDA could both improve as well as impair **comprehension**, especially the readability of individual formulas.

TABLE I
COMMENT DATA RELEVANT TO EXCEL LAMBDAS GATHERED FROM FOUR SOURCES: SUMMARY STATISTICS.

	Words	Comments	Words per comment mean (sd) / median	Threads	Unique user IDs	Date interval (first post - last post)
YouTube	99878	4750	21.03 (30.83) / 13	45	2760	2020/08/13 - 2022/01/27
Microsoft Tech Communities	97363	1446	67.33 (74.61) / 44	182	214	2017/12/10 - 2022/02/01
Hacker News	39206	441	88.90 (130.46) / 47	109	307	2008/04/03 - 2022/01/08
Reddit	18094	362	49.98 (95.22) / 24.5	25	141	2020/12/07 - 2022/01/21
Overall	254541	6999	36.37 (62.27) / 18	361	3422	2008/04/03 - 2022/02/01

[...] sometimes the excel formulas used are very long, convoluted, and hard to grok [...] Many cells often exist as calculation cells only, used for intermediate steps which leads to even more logic complexity. I'm excited to experiment with these to try and simplify some of the long, previously-deemed-necessary calculation methods. [HN_{176}]

If I were to ever find someones sheet using Lambda, chances are that I don't know what are his values refering to [...] [YT_{2021}]

Similarly, commenters anticipated benefits as well as issues in **debugging, testing, and auditing**.

Wow this would make my reports easier to audit as the formula is only defined once. Plus easier [...] to read. [YT_{1264}]

As a financial analyst, my primary worry is that including custom functions will make it hard for another analyst [...] to review my projections [...] without the ability to easily audit a LAMBDA, financial analysts will not adopt this feature. [M_{191}]

Commenters postulated the implications of LAMBDA for the **maintenance** of spreadsheets, in terms of not only the modularity, but also the obfuscation they enabled.

It should make spreadsheets smaller in size, easier to maintain, easier to audit and easier to use. [R_{49}]

I really hope that's true, but there's lots of potential for the exact opposite as people cram entire programs into a single formula. [R_{77}, responding to above.]

In general, commenters were divided on whether LAMBDA functions were the Apollo's Bow ("saving grace") or the "Achilles' Heel" of spreadsheet development.

[...] The saving grace is that the calculation can be hidden within a Named Lambda function [...] which is easier on the eye. [M_5]

[...] cells whose formulas are way too long to be put in a single cell is Excel's Achilles Heel (and a footgun that you are nearly guaranteed to encounter [...]). This LAMBDA proposal as written seems to exacerbate that problem [...]. [HN_9]

To resolve these tensions, commenters proposed **craft practices** [38] for LAMBDA authoring and use.

2) Development of new patterns and practices: Spreadsheet research has already documented the widespread phenomenon of end-users and institutions advocating for development standards and best practices, whether explicitly borrowed from software engineering or independently reinvented. LAMBDA is no exception, prompting discussions and suggestions about patterns and practices when authoring, naming, commenting, and applying it.

Discussions on issues of authoring LAMBDA functions included naming conventions and problem decomposition using LET bindings for subexpressions.

[Referring to a function named COMBINEλ] I added the Greek λ simply as a flag to assist me in reading the formulas [M_5]

the definition of the Lambda function [...] within LET as a locally-scoped function is something I have adopted [M_5]

Others proposed separating error handling and core logic:

[M_{36}] one approach [...] about handling error would be to have Lambda in two layers [...] putting the pseudo code here,

$\mathsf{MAIN} = \mathsf{Lambda}(n, \mathsf{IF}(n <> \mathsf{int}(n), \textit{"Enter integer"}, \mathsf{FIB}(n)))\, \mathsf{FIB} = \mathsf{Lambda}(n, ...)$

Another example is the use of partial application to deal with limited default parameter capabilities:

[M_9] you could approach it like so:

$$\mathsf{Area} := \mathsf{LAMBDA}(dim\_1, \mathsf{LAMBDA}(dim\_2, dim\_1 * dim\_2))$$

Then partially apply the 2, you get:

$$=\mathsf{Area}(2) \Rightarrow =\mathsf{LAMBDA}(dim\_2, 2 * dim\_2)$$

In turn, each of the above proposals sparked questions about their effective use. For example:

to what extent should I use Names for a hierarchy of Lambda functions in order to modularise the code? [M_5]

3) Spreadsheet tooling and experience: Editing and reading long formulas is already challenging [39]. LAMBDA encourages an extended, programmatic style of formula writing that throws into sharp relief the limits of the current formula management environment, exposing several inadequacies of **tooling** and **user experience** for formulas.

Commenters therefore drew comparisons explicitly and implicitly to aspects of tooling well-established in the development environments of professional programmers: syntax highlighting, multi-line editing, parentheses matching, code completions and definitions, and version control.

I don't think name manager cuts it, they'll need a formula manager. Comments between coding lines and tooltips are going to be necessary. Sharing, vital [...] [YT_{1045}]

Functions that were developed and shared during discussions often served a need so fundamental to some user's work that they expressed their desire for its inclusion in the **standard function library**. Often this was motivated by the assumption that a library function would be more performant and comprehensible than a custom function.

Here is my upvote for the SPLIT function. I am very surprised Excel 365 did not have that as a built-in [...] [R_{11}]

You should use native functions where you can. They are faster, more stable, and just better written codes. [...] If you ever need to share your workbook with others, they will appreciate native formula based approach over [custom functions] that they may not understand. [R_{85}]

B. Barriers to learning and using LAMBDA

Commenters encountered barriers that impaired or prevented their use of LAMBDA. We found three out of six programming learning barriers that Ko et al. observed [40]: ***understanding*** barriers, where the LAMBDA behaviour departed from their expectations; *use* barriers, which stopped them authoring and applying LAMBDA; and many instances of ***coordination*** barriers, where users could not combine LAMBDA with other spreadsheet features, such as dynamic arrays. For example:

> *[...] I cannot get even a really simple LAMBDA to work using the Name Manager [...] [M_{37}]*
>
> *The slip was to use the Name [...] within its own definition [...] [M_5, responding to above]*
>
> *What if the LAMBDA function has two variables, then how to use these functions? [YT_{1985}]*
>
> *[...] I wasn't aware at all that the parameter name [...] I chose was actually referring to a very distant cell name in Excel sheet. [...] [M_{26}]*

The causes of barriers ranged from misinterpretation of marketing material, to lack of familiarity with programming jargon that LAMBDA gets it name from, to mistaken assumptions about its capabilities and intended use cases, to the differences between LAMBDA and implementations in other programming languages as anonymous functions.

> *Say I have a simple LAMBDA function := LAMBDA(number, number+1). Is there a way to obtain the "number" by popping out an input box with a prompt like "give me a number"? [M_{14}]*
>
> *you can't use LAMBDAs like VBA to perform interactions or manipulations of things outside the scope of the grid. [M_9, responding to above]*
>
> *By lambda I thought its something related to wavelength of a frequency. [YT_{1828}]*

To address and overcome these barriers, commenters employed a variety of strategies. Some asked for **help**, or searched online documentation. For example:

> *1. Can I save my LAMBDA to Excel so that it can be accessed in all workbooks instead of only the workbook it was created on? 2. When my LAMBDA is called up, can it display the parameters(syntax) the same way a native function does? [M_{35}]*

Others posted examples containing errors and asked others for help in **fixing the error**. In debugging and repairing these errors, commenters were able to get personalised and grounded explanations for aspects of LAMBDA usage.

> *I defined a recursive lambda [...] but received #value. Can u help to resolve this? [M_{21}]*
>
> *Your lambda requires three parameters [...] so your reference to itself should have 3 parameters as well. [M_9, responding to above]*

We observed much **experimentation** and hypothesising. Commenters conducted systematic examinations of certain aspects of LAMBDA usage, such as performance limitations, and interactions with other spreadsheet features, and reported their findings to the community.

> *I don't know if the rest of you love lambdas as much as I do but I have been using them extensively [...] and would like to share one of the tricks I have learned. [...] [R_{19}]*
>
> *I'm trying out excels new LAMBDA function. I'm trying to call it recursively as they write in the press release that you would be able to. [...] [R_{48}]*

Often, commenters were unable to completely determine the reasons for observed behaviours, and in response developed **'folk theories'** [41–43] about LAMBDA, which, while possibly incorrect, facilitated the formation of mental models that allowed commenters to reason about the usage of LAMBDA.

> *[...] If it is actually an iterative function then it is likely to have an underlying recursive function associated with it. The lack of NUM error may be telling or it may simply result from the fact that BYROWS always operates "inside LAMBDA". The more we deal with recursive LAMBDAS the better we will understand their underpinnings. [YT_{1263}]*
>
> *[...] BYROW etc don't seem to have the overhead that recursive functions do. They don't return the NUM! error. I think they've been built using single vectors [...] They know when to stop in the same way as any other formula as they always point to a defined range or array. [...] [YT_{1045}, responding to above]*

C. End-user software engineering

In end-user *programming*, the focus is on completing the task and the program is a means to an end, whereas in end-user *software engineering*, the focus is on the program itself: its correctness, maintainability, and reuse [28]. Discussions of **naming** and **reuse** prompted by LAMBDA signal a shift from the former to the latter.

LAMBDA definitions provoke end-user consideration of naming conventions because, notwithstanding a trivial invocation, names are compulsory arguments to LET and LAMBDA. A key reason for the approachability of spreadsheet programming is the *lack* of a forced consideration of naming: grid references provide automatic names for variables; to instantiate a variable the user needs only enter a value or a formula – the act of choosing a grid location subsumes the act of naming. While assigning names to grid ranges is supported through the name manager, only a small fraction of spreadsheet users know of the feature, let alone use it [44]. Thus, this discussion could only have been precipitated by LET and LAMBDA.

> *[...] Where you've chosen to name your function something related to the report, I think it should be related to the action. [...] with this one and work out the best practice [...] [YT_{1045}]*
>
> *I don't think it is Microsoft's duty to help us name things smartly. That is our duty. It is funny. The vast majority of computer users on the planet earth do not even have the day 1, basic computer skill of naming things smartly. [...] [YT_{1218}]*

Similarly, until the introduction of LAMBDA, users had to learn a different programming language (e.g., VBA, JavaScript) to define functions, and organizational IT teams often restrict their use over security concerns (Section IV-E). The capabilities of LAMBDA leads to speculations about changes to "practice and mindset" required to build spreadsheets.

> *The future is to write robust formulas for complex and reusable tasks [...] or build even more complex ones like subroutines [...] a structural revolution [...] [YT_{370}]*
>
> *After experimenting for a while with LET and now LAMBDA [...] the change of practice and mindset required to create good [...] solutions is so great that one's past experience [...] may itself be the greatest impediment [...] [M_5]*

D. Identity formation

The computational nature of LAMBDA raised discussions regarding the identity of spreadsheet users as programmers, **spreadsheets as code**, and the expertise required to harness LAMBDA. One perspective distinguished spreadsheet authoring and programming, similarly viewing LAMBDA as a separate to coding.

> *That's true if you know VB. Most Excel users don't and this is a simple method to build complex functions without having to code. [YT_{421}]*

> *Many people that use Excel are office workers and don't have enough background in programming or the time to learn it. You can pick up the LAMBDA function in a few minutes though if you are already familiar with many other Excel functions. [YT_{421}]*

Prior work suggests that this perspective can be motivated by wanting to distance oneself from a "programmer" [45]. This tension of identity was also reflected in the discussion.

> *Whilst [...] Lambda functions are simple extensions of the traditional spreadsheet methods [...] the process of solution development shifts from the ad-hoc to being a programming exercise [...] traditional methods allow users to interact with their numbers whilst remaining in denial that they are actually programming [M_5]*

The **expertise** required to use LAMBDA was another common theme. Central to the discussion was the notion of expertise across an organization, particularly the practice of those with expertise building and **sharing** solutions with those who do not. Some viewed LAMBDA as another advanced feature reserved for those with expertise.

> *Most organizations [...] only have a couple people who can actually make and edit the advanced sheets [...] and lots of people who use those sheets with a very, very rudimentary knowledge of Excel to generally get their jobs done. I don't really see the addition of new advanced functionality changing that [...] [HN_{38}]*

Several users expressed reluctance or concerns because collaborators might not be familiar with LAMBDA functions; thus, LAMBDA functions might hinder comprehension.

> *[...] though how might non-technical folks feel about a coworker sending them a spreadsheet with function-values? [HN_4]*

In contrast, others indicated that LAMBDA can improve sharing by providing a way to implement advanced functionality without resorting to traditional programming such as VBA, reaffirming that some users perceive a distinction between spreadsheets and code.

> *There's always been this midpoint of complexity when making spreadsheets that coworkers will use. Including macros scares them away but using long, un-named formulas does not even though it is much less clear what it's doing [...] a macro was not needed, only a label and some arguments to lower the cognitive load. [HN_{95}]*

Consistent with prior observations, we observed derision and gatekeeping behaviour regarding the identity and expertise of spreadsheet users as programmers [45].

> *What has instead happened, I shit you not, is we have "Excel influencers" teaching people recursion "without code!" (aka without VBScript). I cried and I laughed when I first saw, it was a watershed moment in CS education. [HN_{48}]*

> *[...] no mention of debugging. In the hands of Excel cowboys, this can become another foot gun. [HN_{124}]*

> *[...] The irony is that this feature will almost certainly be met with derision & scorn from the CS crowd and clueless shrugs from excel users. [HN_{182}]*

However, we also observed the championing and support for the perspective that spreadsheet users are part of a programming community.

> *Of course this also means that Excel becomes even more of a slippery slope towards programming generally, which can only be a good thing. [HN_{161}]*

> *[...] this could actually be a really natural bridge into programming for a lot of people whose advanced Excel skills already have them on the cusp. [HN_{119}]*

Debate about the identity of spreadsheet users as programmers is by no means new [45], but the introduction of a first-class computational abstraction is a fresh opportunity for users and researchers alike to revisit this discussion.

E. Alternatives

Discussions of potential **alternatives** to LAMBDA, including other ways of achieving the same thing as LAMBDA, or simply citations of other technologies, helped users situate LAMBDA among a wider suite of tools. The most common comparison was to VBA. Users discussed other programming languages, software packages, and alternative ways of using existing spreadsheet features.

After expertise, which we have covered, the salient axes of comparison were **security** and **performance**. When contrasted with VBA macros, LAMBDA was seen to have security and performance benefits, due to the perceived superiority of the formula calculation engine over the VBA environment.

> *Alternatively, Excel lambda-based udfs will be faster than VBA UDFs because the Excel Calc engine is multi-threaded, whereas VBA is single-threaded. [R_{21}]*

> *Lambdas also have the advantage of working in situations where macros are not allowed or on platforms where VBA is not available [...] though the latter also allows you to perform many workbook management tasks. [YT_{909}]*

> *The problem with "macros" is that they can be arbitrary VBA code that can invoke OS functions and foreign applications. Lambdas can only invoke Excel functions that you can invoke anyway [...] Lambdas merely add an abstraction mechanism, they otherwise don't provide access to new functionality. [HN_{88}]*

F. Differences in communities

We observed differences between the four communities studied, reflective of the differing demographics of users of these websites and their differing concerns. Together, they showcase a diverse range of stakeholder perspectives.

For example, Hacker News, a community aimed towards technology industry professionals, involved comparative and analytical comments that considered the reception and consequences of LAMBDA on the spreadsheet community. Comments from Hacker News were the longest on average (median=47 words), in part because they explored technical concepts. Many drew comparisons with programming concepts, research that inspired LAMBDA, potential improvements based

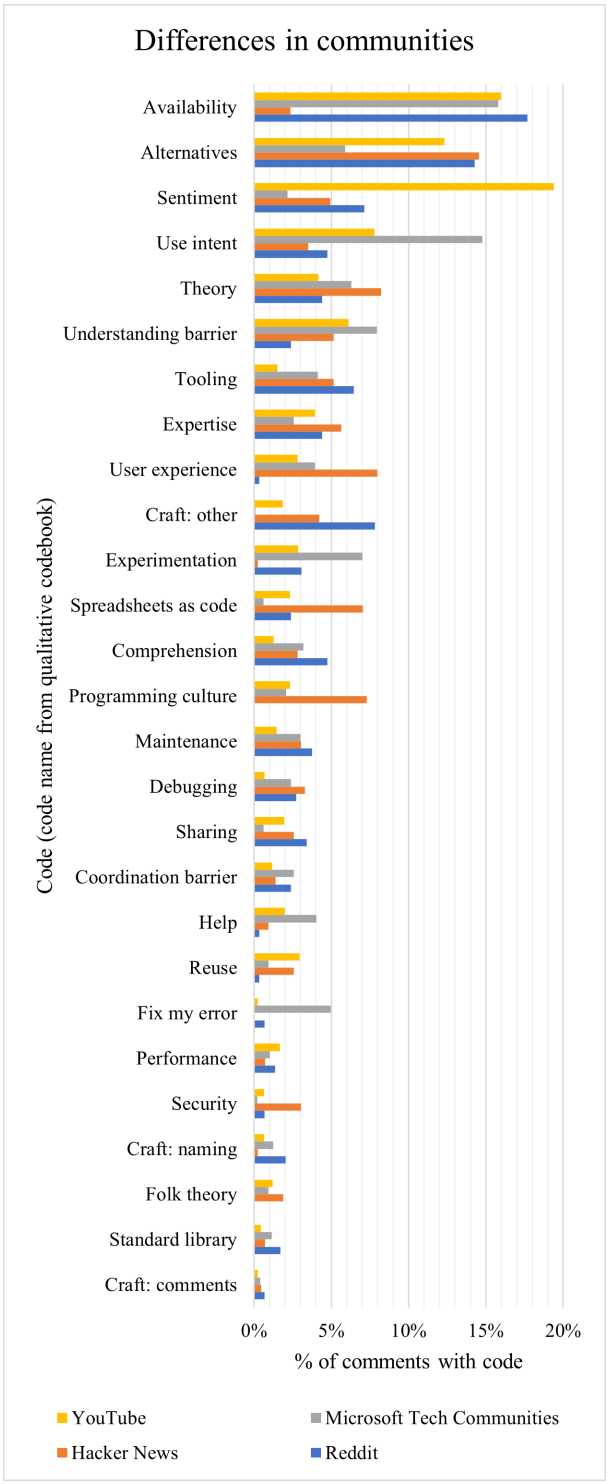

Fig. 1. Relative frequencies of qualitative codes, compared between different online communities.

on other programming paradigms, or whether using spreadsheets constituted programming.

On the other hand, Microsoft Tech Community is the official discussion and help forum for Microsoft products, including Excel. Users fell largely into two groups: a small number of experts who contributed to several threads, and a large number of users who came to seek help, started a single thread, and posted only a few comments on it. Comments were somewhat long (median=44 words) and often included formulas and attached spreadsheets. This can be seen in the high occurrence of **use intent** and **experimentation**, both of which relate to users writing examples of LAMBDA (Fig. 1).

Reddit discussions were more concerned with sharing **craft** practices. Most threads came from the `r/excel` forum, which is focused on sharing techniques and news, and oriented towards practitioners and enthusiasts. Reddit comments discussed naming and commenting, how to use LAMBDA to improve maintainability and comprehension, and pitfalls to avoid. Reddit users appeared to have a wide range of expertise. Some had formal knowledge of programming, although a smaller proportion than Hacker News.

YouTube comments were often in response to tutorial videos, which users presumably watched to learn about LAMBDA. Our YouTube data contained more unique users and comments than the other three sources combined, but its comments were also the shortest (median=13 words), and often not informative. YouTube comments often reflected users' unjustified **sentiment** towards LAMBDA (Fig. 1).

To illustrate the differences, we present the relative frequency of codes across different communities in Figure 1. We recognise inherent limitations in attempting to quantify qualitative codes, and present this purely as an narrative aid. We are not making quantitative claims about the representativeness of these code frequencies.

V. Discussion

A. *Are spreadsheet users programmers?*

The question of whether spreadsheet users are programmers is not inconsequential terminological hair-splitting; it has implications for the role spreadsheets play in society.

Programmer is a marked identity [46] with cultural associations against which end-users judge themselves. The conceptualisation of spreadsheet use as a programming activity therefore directly suggests to users whether spreadsheets are for people 'like them' or for 'others'. People's skills are shaped and captured by software, resulting in an intimate binding of one's professional value to the software one uses [47].

It has been previously observed that spreadsheet users do not view themselves as programmers [45]. Our findings provide additional nuance: some spreadsheet users view themselves as programmers, but others do not. Some are unaware that a distinction can be made, or if a distinction is to be made, what the appropriate grounds are. Users who concede that spreadsheet formula authoring can be viewed as programming may nonetheless not self-identify as programmers.

boyd and Ellison describe the community process of *impression management* [48], where the expression of self through social media becomes a mechanism for forming and signalling identity. Though highly individualised, the identity of "programmer" is at the same time a community identity, subject to community negotiation. In this respect it is similar to sociological theories of group formation [49]. Kinship, friendship,

and neighbourly groups serve as means of allocating resources, but also as a psychological device for uniting the sense of self with a sense of belonging. Thus, the use and discussion of LAMBDA can serve both as a peer-signalling mechanism as well as to enact one's own identity as a programmer.

LAMBDA creates new bridges (and gulfs) between programmers and 'others'. Many commenters implied that using LAMBDA is closer to traditional notions of programming than authoring spreadsheet formulas, but it is unclear what underpins this perception. Does managing complexity via LAMBDA resemble software engineering practices, associated with programming and dissociated from spreadsheets? As Wing notes, decomposition and abstraction are key to computational thinking [50].

Some commenters predicted that LAMBDA is going to change the way spreadsheets are built. However, absent better tooling and higher levels of user expertise, it is doubtful whether a LAMBDA-centric spreadsheet development approach induces better design, testing, or documentation. While some users anticipated a "structural revolution", such optimism ought to be tempered by the fact that formula writers constitute a minority of spreadsheet users, and corpus studies show that as few as 7% of spreadsheets contain a single formula [51, 52].

B. Abstraction and comprehension

Ragavan et al.'s study of spreadsheet comprehension found two main bottlenecks to formula comprehension: first, the information-seeking detours required to understand the quantities on which a formula operates; and second, understanding the usage of unfamiliar formula functions [39].

While some commenters felt that LAMBDA improved formula comprehension, particularly regarding 'megaformulas', others felt that these problems would be worsened. Ragavan et al.'s findings allow us to identify the source of this contradiction: first, the naming of subexpressions can either eliminate or necessitate information-seeking detours; and second, with LAMBDA, there is both the potential for more intelligible, domain-specific functions as well as for arbitrary, poorly-documented constructions. The resolution of this tension cannot be (purely) in the technical design of the formula language, and thus we observed users develop craft practices in response.

C. Implications for spreadsheets and other tools

LAMBDA promotes attention investment in complex formulas, and thus raises issues of tooling. Discussions made clear the value of ideas from professional development environments, such as syntax highlighting, code definitions, autocomplete, versioning and library management, and so on. We are not the first to observe such needs or propose their implementation. However the wholesale implementation of professional features for end-user programmers is seldom an effective strategy; the design of Calculation View [44] shows how a code-like representation might be sympathetically introduced as a companion to the spreadsheet grid.

Opportunities also arise for misuse. Common anti-patterns in programming (e.g., abuse of recursion, deeply nested conditions, deep chains of invocations) may surface through LAMBDA. Opportunities emerge for educating spreadsheet users about such misuses, and for addressing them (e.g., via refactoring tools [53]). By analogy to prior research, there are research opportunities in unit testing and documentation [28].

D. Limitations

Analyses of online communities are subject to self-selection bias; participation in our dataset is skewed towards end-user programmers with an intrinsic interest in LAMBDA and in developing their technical skills. This group may not be representative. On the other hand, attitudes and practices of spreadsheet use often percolate through the end-user community via influential, opinionated, and expert individuals who advocate for features and techniques [54, 55]. As such, it seems advantageous to study a group whose views are likely to influence others.

During coding, we felt limited by our *unitisation* of the data into individual comments. Sometimes codes were more properties of (sub-)threads than of individual comments. There were digressive and discursive comments which, despite arising in the context of one set of codes, were not directly relevant to them. We developed heuristics for consistency (e.g., to qualify for a code, a reply must add new information and not merely repeat or agree with a previous comment) but still felt the need for a more flexible scheme than applying codes to individual comments. While in this study we followed the single-comment model of previous work [33], this strikes us as an area with potential for methodological innovation.

Our data captures an early stage of the LAMBDA release, with access limited for most of this period to a public beta testing program (albeit a very large one, with millions of members). We could have waited until LAMBDA had been in general release for some time. However, observations of user experiences at all stages of a product lifecycle are useful; documenting early barriers and misconceptions will enable comparisons with user attitudes in the future.

VI. Conclusion and future work

We studied reactions to lambda abstractions in spreadsheets through a thematic analysis of nearly 2,700 comments posted on four online forums. Commenters noted that abstraction has both benefits and drawbacks, and that complementary practices (e.g., variable naming, documentation) and better tooling (e.g, editors, sharing and versioning tools, testing and debugging utilities) are necessary to use LAMBDA effectively. They also anticipated challenges around comprehension by less-expert users. LAMBDA prompted mindset changes towards investing in authoring more reusable abstractions.

The Excel LAMBDA function reopens debate around the identity of spreadsheet users as programmers. It raises questions of whether users ought to learn a 'traditional' programming language to access concepts such as abstraction and recursion, and whether lambda abstractions are a smooth segue for spreadsheet users to begin learning a language such as Python. Each of these opens a rich avenue for future work.

APPENDIX
CODES AND THEIR DESCRIPTIONS

- **irrelevant**: not related to lambdas, or related to lambdas but not informative, or not in English

A. The tensions of abstraction

- **comprehension**: All discussions about ability to understand/read/comprehend formulas/lambdas (or the lack), granular at the level of a single formula
- **debugging, testing, auditing**: (abstract) The ability to debug, test, audit formulas/lambdas, their ux, challenges
- **maintainability**: discussions about formula and spreadsheet maintainability made better or worse (includes simplification, modularity, error-proneness), only if this doesn't fall under debugging testing and auditing. Discussions about code-quality (code being formula), importance, or lack thereof
- **craft practices: naming**: discussion about naming of parameters, the challenges, importance, etc
- **craft practices: comments**: discussion about the ability to add comment in formulas
- **craft practices: others**: other craft practices; suggest some usage idiom or technique
- **availability**: discussion about access to LAMBDA, channels, Excel versions, various end points (e.g., desktop, web, and mobile)
- **sharing**: sharing formulas and lambdas, the challenges; a reuse scenario where colleagues/other people are explicitly mentioned, or sharing files with lambdas / collaboration (does not include sharing files in a forum for debugging support)
- **reuse**: reusing formulas and lambdas, across workbooks, from coworkers; mentions scenarios where lambda is used multiple times

B. Implications for tools and practices

- **user experience**: consideration of aspects of the experience of overall current lambda usage, but not mentioning tooling keywords (e.g., current syntax, current way of writing)
- **tooling**: explicit mention of: editor, version control, IDE, debugger, package manager, repository. A subset of user experience.
- **standard formula library**: references to functionality that should be built in to spreadsheet formula language / functions, e.g. regex
- **performance**: discuss speed, memory requirements, etc. for lambdas (or alternatives, in comparison)
- **security**: discuss security issues for lambdas (or alternatives, in comparison)

C. Barriers and strategies

- **understanding barriers**: the process of understanding lambdas, the desire to, the challenge, common misunderstandings. Also, misreadings - perhaps assumptions based on marketing material about what lambdas are like. Includes asking questions about lambda understanding. This also includes 'understanding' and 'use' barriers as per Ko et al.'s framework.
- **coordination barrier**: any confusion or remark about interaction between *lambdas* and some other Excel feature
- **sentiment**: Positive / negative sentiment about lambdas with no justifications (e.g., lambdas are nice).
- **use intent or scenario**: shares concrete current or anticipated use cases for lambdas (or a use case inspired by discussion of lambdas, even if lambdas are not the best tool to solve the problem)
- **sensemaking: help**: use of help resources, their usefulness, or their unhelpfulness. Asking others on the internet for help on things that should be in documentation
- **sensemaking: experimentation**: 'I tried/will try X', probing the behaviour and possibilities of lambda
- **folk theory**: guessing about lambda behaviour, or why lambdas interact with other features in a certain way (without actually trying something out, or trying but not getting direct evidence for the guess). Specifically refers to an unverified guess or hypothesis about how lambdas work. Look out for sentences such as: 'I think what is happening is X', or 'I guess X happens becuse Y'. An episode of understanding (non-barrier) (e.g., 'I don't understand why Y happens') may culminate in folk (e.g., 'I guess it is because of X') if it isn't or can't be properly answered in the thread.
- **fix my error**: comments that are requesting or attempting to fix a bug in a lambda

D. Identity formation

- **spreadsheets as code**: discussion about the differences/similarities of spreadsheets and programming, whether users are programmers, "real programming"
- **spreadsheet expertise**: consideration of skill/knowledge required to use lambdas or alternatives
- **programming theory**: explicit mention of formal academic concepts from computer science or programming theory such as types, currying, recursion
- **programming culture**: references to programming culture, or concepts adjacent to programming, such as project-euler, XKCD, scheme, lisp interpreter, or tracing lineage of functional programming ideas (overlaps with alternatives, if they mention concrete implementations)
- **alternatives**: discussions of potential alternatives to LAMBDA (including other ways of doing the same thing as LAMBDA), or comparison/citation of other technologies (thus helping situate Lambda among wider suite of tools)

Acknowledgments

We thank the reviewers for their careful feedback.

References

[1] B. Jones, "Announcing LAMBDA: Turn Excel formulas into custom functions," Dec 2020. [Online]. Available: https://techcommunity.microsoft.com/t5/excel-blog/announcing-lambda-turn-excel-formulas-into-custom-functions/ba-p/1925546

[2] A. Gordon and S. Peyton Jones, "LAMBDA: The ultimate Excel worksheet function," Jan. 2021. [Online]. Available: https://www.microsoft.com/en-us/research/blog/lambda-the-ultimatae-excel-worksheet-function/

[3] R. Munroe, "Excel Lambda," 2021. [Online]. Available: https://m.xkcd.com/2453/

[4] R. Rojas and U. Hashagen, "Konrad Zuse's Z4: architecture, programming, and modifications at the ETH Zurich," 2002.

[5] B. E. Carpenter and R. W. Doran, "The other Turing machine," *The Computer Journal*, vol. 20, no. 3, pp. 269–279, 01 1977. [Online]. Available: https://doi.org/10.1093/comjnl/20.3.269

[6] M. Burnett, J. Atwood, R. W. Djang, J. Reichwein, H. Gottfried, and S. Yang, "Forms/3: A first-order visual language to explore the boundaries of the spreadsheet paradigm," *Journal of functional programming*, vol. 11, no. 2, pp. 155–206, 2001.

[7] S. Peyton Jones, A. Blackwell, and M. Burnett, "A user-centred approach to functions in Excel," in *Proceedings of the eighth ACM SIGPLAN international conference on Functional programming*, 2003, pp. 165–176.

[8] M. Mccutchen, J. Borghouts, A. D. Gordon, S. Peyton Jones, and A. Sarkar, "Elastic sheet-defined functions: Generalising spreadsheet functions to variable-size input arrays," *Journal of Functional Programming*, vol. 30, 2020.

[9] G. Chalhoub and A. Sarkar, ""It's Freedom to Put Things Where My Mind Wants": Understanding and Improving the User Experience of Structuring Data in Spreadsheets," in *Proceedings of the 2022 CHI Conference on Human Factors in Computing Systems*, ser. CHI '22. New York, NY, USA: Association for Computing Machinery, 2022.

[10] N. Joharizadeh, A. Sarkar, A. D. Gordon, and J. Williams, "Gridlets: Reusing spreadsheet grids," in *Extended Abstracts of the 2020 CHI Conference on Human Factors in Computing Systems*, 2020, pp. 1–7.

[11] A. F. Blackwell, L. Church, and T. R. Green, "The abstract is an enemy: Alternative perspectives to computational thinking." in *PPIG*, 2008, p. 5.

[12] A. Church, "A set of postulates for the foundation of logic," *Annals of Mathematics*, vol. 33, no. 2, pp. 346–366, 1932.

[13] ——, *The calculi of lambda-conversion*. Princeton University Press, 1941.

[14] P. J. Landin, "Correspondence between ALGOL 60 and Church's lambda-notation: parts I and II," *Communications of the ACM*, vol. 8, no. 1–2, 1965.

[15] G. L. Steel Jr. and G. J. Sussman, "LAMBDA: The ultimate imperative," MIT AI Laboratory, AI Memo 353, Mar. 1976. [Online]. Available: https://dspace.mit.edu/handle/1721.1/5790

[16] T. R. G. Green and M. Petre, "Usability analysis of visual programming environments: a 'cognitive dimensions' framework," *Journal of Visual Languages & Computing*, vol. 7, no. 2, pp. 131–174, 1996.

[17] A. F. Blackwell, "Psychological issues in end-user programming," in *End user development*. Springer, 2006, pp. 9–30.

[18] B. Shneiderman, "Direct manipulation for comprehensible, predictable and controllable user interfaces," in *Proceedings of the 2nd international conference on Intelligent user interfaces*, 1997, pp. 33–39.

[19] A. F. Blackwell, "See what you need: Helping end-users to build abstractions," *Journal of Visual Languages & Computing*, vol. 12, no. 5, pp. 475–499, 2001.

[20] D. Kurlander, A. Cypher, and D. C. Halbert, *Watch what I do: programming by demonstration*. MIT press, 1993.

[21] S. Gulwani, "Automating string processing in spreadsheets using input-output examples," *ACM Sigplan Notices*, vol. 46, no. 1, pp. 317–330, 2011.

[22] P. Sestoft and J. Z. Sørensen, "Sheet-defined functions: implementation and initial evaluation," in *International Symposium on End User Development*. Springer, 2013, pp. 88–103.

[23] J. Williams, N. Joharizadeh, A. D. Gordon, and A. Sarkar, "Higher-order spreadsheets with spilled arrays." in *ESOP*, 2020, pp. 743–769.

[24] M. Burnett, "What is end-user software engineering and why does it matter?" in *International symposium on end user development*. Springer, 2009, pp. 15–28.

[25] M. F. Costabile, D. Fogli, P. Mussio, and A. Piccinno, "End-user development: The software shaping workshop approach," in *End user development*. Springer, 2006, pp. 183–205.

[26] T. A. Grossman, "Spreadsheet engineering: A research framework," *arXiv preprint arXiv:0711.0538*, 2007.

[27] J. Cunha, M. Erwig, and J. Saraiva, "Automatically inferring ClassSheet models from spreadsheets," in *2010 IEEE Symposium on Visual Languages and Human-Centric Computing*. IEEE, 2010, pp. 93–100.

[28] A. J. Ko, R. Abraham, L. Beckwith, A. Blackwell, M. Burnett, M. Erwig, C. Scaffidi, J. Lawrance, H. Lieberman, B. Myers *et al.*, "The state of the art in end-user software engineering," *ACM Computing Surveys (CSUR)*, vol. 43, no. 3, pp. 1–44, 2011.

[29] R. R. Panko and D. N. Port, "End user computing: The dark matter (and dark energy) of corporate IT," *Journal of Organizational and End User Computing (JOEUC)*, vol. 25, no. 3, pp. 1–19, 2013.

[30] B. A. Nardi and J. R. Miller, "Twinkling lights and

nested loops: distributed problem solving and spreadsheet development," *International Journal of Man-Machine Studies*, vol. 34, no. 2, pp. 161–184, 1991.

[31] Y. Wu, J. Kropczynski, P. C. Shih, and J. M. Carroll, "Exploring the ecosystem of software developers on github and other platforms," in *Proceedings of the Companion Publication of the 17th ACM Conference on Computer Supported Cooperative Work & Social Computing*, ser. CSCW Companion '14. New York, NY, USA: Association for Computing Machinery, 2014, p. 265–268. [Online]. Available: https://doi.org/10.1145/2556420.2556483

[32] T. Barik, B. Johnson, and E. Murphy-Hill, "I heart hacker news: expanding qualitative research findings by analyzing social news websites," in *Proceedings of the 2015 10th Joint Meeting on Foundations of Software Engineering*, 2015, pp. 882–885.

[33] T. Barik, "Expressions on the nature and significance of programming and play," in *2017 IEEE Symposium on Visual Languages and Human-Centric Computing (VL/HCC)*. IEEE, 2017, pp. 145–153.

[34] V. Braun and V. Clarke, "Using thematic analysis in psychology," *Qualitative research in psychology*, vol. 3, no. 2, pp. 77–101, 2006.

[35] J. L. Campbell, C. Quincy, J. Osserman, and O. K. Pedersen, "Coding in-depth semistructured interviews: Problems of unitization and intercoder reliability and agreement," *Sociological methods & research*, vol. 42, no. 3, pp. 294–320, 2013.

[36] N. McDonald, S. Schoenebeck, and A. Forte, "Reliability and inter-rater reliability in qualitative research: Norms and guidelines for CSCW and HCI practice," *Proceedings of the ACM on Human-Computer Interaction*, vol. 3, no. CSCW, pp. 1–23, 2019.

[37] J. Saldaña, *The coding manual for qualitative researchers*. sage, 2021.

[38] A. Blackwell, "A craft practice of programming language research," in *Proceedings of the Psychology of Programming Interest Group (PPIG) Conference*, 2018.

[39] S. Srinivasa Ragavan, A. Sarkar, and A. D. Gordon, "Spreadsheet comprehension: Guesswork, giving up and going back to the author," in *Proceedings of the 2021 CHI Conference on Human Factors in Computing Systems*, ser. CHI '21. New York, NY, USA: Association for Computing Machinery, 2021. [Online]. Available: https://doi.org/10.1145/3411764.3445634

[40] A. J. Ko, B. A. Myers, and H. H. Aung, "Six learning barriers in end-user programming systems," in *2004 IEEE Symposium on Visual Languages-Human Centric Computing*. IEEE, 2004, pp. 199–206.

[41] P. N. Johnson-Laird and K. Oatley, "Basic emotions, rationality, and folk theory," *Cognition & Emotion*, vol. 6, no. 3-4, pp. 201–223, 1992.

[42] M. Eslami, K. Karahalios, C. Sandvig, K. Vaccaro, A. Rickman, K. Hamilton, and A. Kirlik, "First I "like" it, then I hide it: Folk Theories of Social Feeds," in *Proceedings of the 2016 CHI conference on human factors in computing systems*, 2016, pp. 2371–2382.

[43] M. A. DeVito, J. Birnholtz, J. T. Hancock, M. French, and S. Liu, "How people form folk theories of social media feeds and what it means for how we study self-presentation," in *Proceedings of the 2018 CHI conference on human factors in computing systems*, 2018, pp. 1–12.

[44] A. Sarkar, A. D. Gordon, S. Peyton Jones, and N. Toronto, "Calculation view: multiple-representation editing in spreadsheets," in *2018 IEEE Symposium on Visual Languages and Human-Centric Computing (VL/HCC)*, 2018, pp. 85–93.

[45] F. Hermans, B. Jansen, S. Roy, E. Aivaloglou, A. Swidan, and D. Hoepelman, "Spreadsheets are code: An overview of software engineering approaches applied to spreadsheets," in *2016 IEEE 23rd International Conference on Software Analysis, Evolution, and Reengineering (SANER)*, vol. 5. IEEE, 2016, pp. 56–65.

[46] W. Brekhus, "A sociology of the unmarked: Redirecting our focus," *Sociological Theory*, vol. 16, no. 1, pp. 34–51, 1998.

[47] M. Nouwens and C. N. Klokmose, "The application and its consequences for non-standard knowledge work," in *Proceedings of the 2018 CHI Conference on Human Factors in Computing Systems*, 2018, pp. 1–12.

[48] d. m. boyd and N. B. Ellison, "Social network sites: Definition, history, and scholarship," *Journal of computer-mediated Communication*, vol. 13, no. 1, pp. 210–230, 2007.

[49] E. Litwak and I. Szelenyi, "Primary group structures and their functions: Kin, neighbors, and friends," *American Sociological Review*, pp. 465–481, 1969.

[50] J. M. Wing, "Computational thinking," *Communications of the ACM*, vol. 49, no. 3, pp. 33–35, 2006.

[51] F. Hermans and E. Murphy-Hill, "Enron's spreadsheets and related emails: A dataset and analysis," in *2015 IEEE/ACM 37th IEEE International Conference on Software Engineering*, vol. 2. IEEE, 2015, pp. 7–16.

[52] T. Barik, K. Lubick, J. Smith, J. Slankas, and E. Murphy-Hill, "FUSE: a reproducible, extendable, internet-scale corpus of spreadsheets," in *2015 IEEE/ACM 12th Working Conference on Mining Software Repositories*. IEEE, 2015, pp. 486–489.

[53] F. Hermans, M. Pinzger, and A. van Deursen, "Detecting and refactoring code smells in spreadsheet formulas," *Empirical Software Engineering*, vol. 20, no. 2, pp. 549–575, 2015.

[54] A. Sarkar and A. D. Gordon, "How do people learn to use spreadsheets? (work in progress)," in *Proceedings of the 29th Annual Conference of the Psychology of Programming Interest Group (PPIG 2018)*, 2018, pp. 28–35.

[55] J. C. Brancheau and J. C. Wetherbe, "The adoption of spreadsheet software: testing innovation diffusion theory in the context of end-user computing," *Information systems research*, vol. 1, no. 2, pp. 115–143, 1990.