The Effect of a Block-based Language on Formula Comprehension in Spreadsheets

Jansen, Bas; Hermans, Felienne

Important note
To cite this publication, please use the final published version (if applicable).
Please check the document version above.

The Effect of a Block-based Language on Formula Comprehension in Spreadsheets

Bas Jansen
Delft University of Technology
The Netherlands
Email: b.jansen@tudelft.nl

Felienne Hermans
Leiden University
The Netherlands
Email: f.f.j.hermans@liacs.leidenuniv.nl

Abstract—The use of spreadsheets in industry is widespread. It is known that spreadsheets have an average life span of five years, and during this life span, they are used on average by thirteen different persons. Consequently, spreadsheets need maintenance, and knowledge about the spreadsheet needs to be transferred from one user to another. To minimize the risk of introducing new errors, a thorough understanding of the spreadsheet's formulas is needed during maintenance and knowledge transfer tasks.

Research on the use of block-based languages has shown that they positively affect the comprehension of program code. We hypothesize that using a block-based representation of a spreadsheet formula will positively affect formula comprehension.

Hence, we extended XLBlocks, a block-based formula editor for spreadsheets, with the functionality to generate a block-based representation of an existing formula. We conduct a think-aloud study with twenty-one experienced spreadsheet users from industry and ask them to perform a set of spreadsheet comprehension tasks using XLBlocks. During an interview, we ask them, using the Cognitive Dimensions of Notations framework, to reflect on the use of XLBlocks.

We found that participants preferred to use the block-based representation of formulas when analyzing or explaining formulas or to implement non-trivial changes. Named function parameters and the absence of parentheses and commas make functions easier to understand. Furthermore, the visualization enables the user to separate smaller parts in the formula, which improves comprehension. Finally, the possibility to navigate from formula to formula makes it clear how formulas work together and improve the understanding of the spreadsheet as a whole.

I. INTRODUCTION

Spreadsheets are ubiquitous in industry and often used for critical business decisions. Unfortunately, spreadsheets are also known for their error-proneness. Almost all spreadsheets contain non-trivial errors [1]. Consequently, companies are at risk of basing their decisions on inaccurate information, which can lead to significant loss of money or reputation.[1]

A major part of spreadsheet research is focused on improving spreadsheets by applying software engineering methods. For example, the concept of testing in spreadsheets was studied by Rothermel *et. al.* [2] and more recently by Roy *et. al.* [3]. Hermans *et. al.* [4] and Cunha *et. al* [5] introduced the idea of reverse engineering of spreadsheets and designed methods for extracting class diagrams from spreadsheets. Several studies [6] [7] [8] define and investigate code smells in spreadsheets. Refactoring is closely related to code smells, and both Badame

and Dig [9] and Hermans and Dig [10] developed tools for refactoring in spreadsheets.

The common denominator in these studies is that they provide methods and techniques that support users in improving spreadsheets. Nevertheless, a focus on spreadsheet comprehension is lacking. According to Hermans *et. al.* [11], spreadsheets have an average life span of five years and are on average used by thirteen different users. This means that during a spreadsheet's lifetime, maintenance is needed, and for that, knowledge needs to be transferred from one user to another. During these 'transfer scenarios', a thorough understanding of the spreadsheet minimizes the risk of introducing new errors.

Therefore, we focus in this paper on formula comprehension. In an earlier study [12] we introduced XLBlocks, a block-based formula editor for spreadsheets. With this editor, it is possible to create formulas with a block-based language instead of the default textual formula language and translate them automatically into valid spreadsheet formulas. However, in our first implementation of XLBlocks, it was impossible to generate a block-based representation from a formula, making it less suitable for formula comprehension. For this study, we have extended XLBlocks with the functionality to generate from a textual formula a block-based representation of that formula. This enables users to analyze existing spreadsheet formulas in a block-based language.

In this paper, we want to understand the effect of a block-based language for spreadsheets on formula comprehension. To answer this question, we conduct a think-aloud study in which we ask participants to perform a set of formula comprehension tasks with a new version of XLBlocks. When they have completed these tasks, we interview them and ask them to reflect on their experience with XLBlocks. To guide the interview, we use the Cognitive Dimensions of Notation (CDN) Framework [13].

II. RELATED WORK

A. Spreadsheets and Visual Languages

Related to our study is the work of Burnet *et. al.* [14]. They introduced the visual research language Forms/3. This language's goal was to eliminate some of the spreadsheet systems' limitations without abandoning the spreadsheet paradigm. The language describes a complete spreadsheet

---

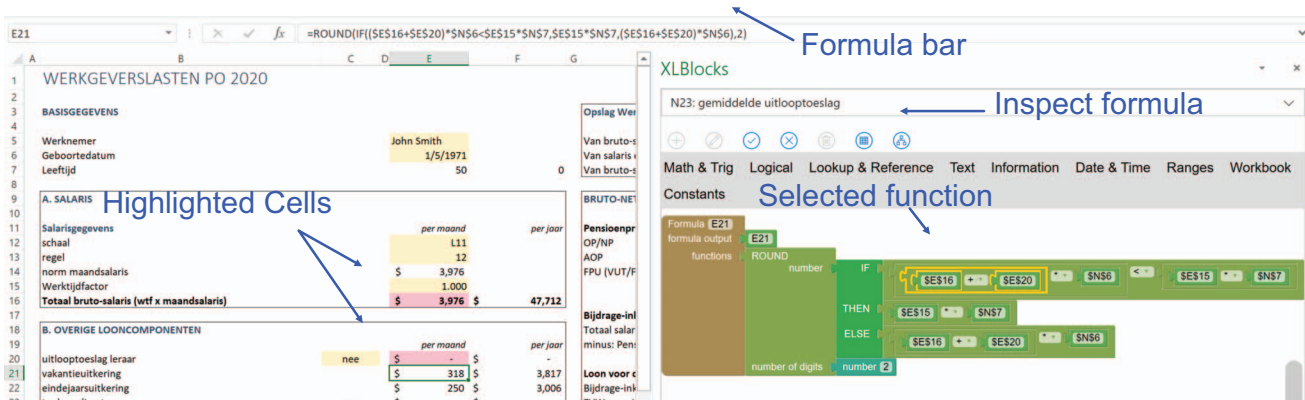[1] http://www.eusprig.org/horror-stories.htm

Fig. 1. Left the traditional view with the formula bar at the top, right the XLBlocks interface showing the block representation of the formula

model. This in contrast to XLBlocks, where we focus on an individual formula.

Abraham *et. al.* [15] also introduced a visual language for spreadsheets called ViTSL. With ViTSL, it is possible to define a spreadsheet template. From such a template, a spreadsheet can be generated automatically. Based on this work, Engels and Erwig [16] introduced ClassSheet. A ClassSheet represents both the structure and the relation between (business) objects within the spreadsheet. With ClassSheets, the problem domain and spreadsheet domain are brought closer together. The ClassSheets needed to be developed in a stand-alone application, and for that reason, real-time synchronization between the ClassSheet and the spreadsheet was not possible. Cunha *et. al.* [17] integrated ClassSheets in the spreadsheet and enabled two-way synchronization between the ClassSheet and the spreadsheet.

Leitão and Roast [18] developed a visual language for formulas. It is not a block-based language but a data-flow language. They worked on two different variants: Explicit Visualization (EV) and Data flow Visualization (DV). In the EV, the visualization is a direct match of the spreadsheet formula. Operators, cell references, constants, and functions have been replaced by symbols. The DV uses the same symbols, but they are presented as a syntax tree.

Finally, Sarkar et al. [19] introduced Calculation View, which is also an alternative representation of the spreadsheet. Formulas are presented in a textual calculation view adjacent to the standard grid view. One of Calculation View features is 'range assignments', which allows the user to assign the same formula to a range of cells. This is more efficient and less maintenance intensive than manually dragging the formula down to a range of cells. Furthermore, with Calculation View, it is possible to name cells easily and refer to those names in other formulas.

B. Block-based languages

BLOX can be considered the first block-based language and was introduced by Glinert [20]. After the introduction of several block-based languages, like Alice [21], Scratch [22], and Blockly [23] the body of research on block-based languages started to grow.

Most related to our research is a study of Weintrop *et. al.* [24]. In this study, they introduce CoBlox, a block-based language to program a one-armed industrial robot. They demonstrate that block-based languages are not only suitable for children in an educational environment but also useful for adult novice programmers in an industrial setting. Adult programmers successfully implemented robot programs with CoBlox faster and with no loss in accuracy than similar programmers using one of two widely-used industrial robot programming approaches. They also scored better on usability, learnability, and overall satisfaction.

Weintrop *et. al.* also conducted a study on block-based comprehension [25]. They asked participants to answer twenty program comprehension questions using two variants (text-based and block-based) of pseudocode developed for the Advanced Placement CS Principles course. They concluded that learners performed better on questions presented in the block-based modality.

Finally, Hermans and Holwerda [26] conducted a user study with ArduBlockly. Using a user study, they demonstrate a usability analysis of block-based editors based on the Cognitive Dimensions of Notation (CDN) framework. Furthermore, they give an overview of several design maneuvers to improve programming time and effort, program comprehension, and programmer comfort.

III. ANALYZING FORMULAS WITH XLBLOCKS

A. XLBlocks Interface

Figure 1 shows the XLBlocks interface. The spreadsheet is displayed on the left side of the screen, the XLBlocks editor on the right side. At the top of the screen is the formula bar, which is the Excel default tool to enter formulas. A formula can be analyzed in XLBlocks by selecting the cell with the formula and using the inspect formula button to generate the formula's block-based representation. A video demonstrating the user interface of XLBlocks is available on-line.[2]
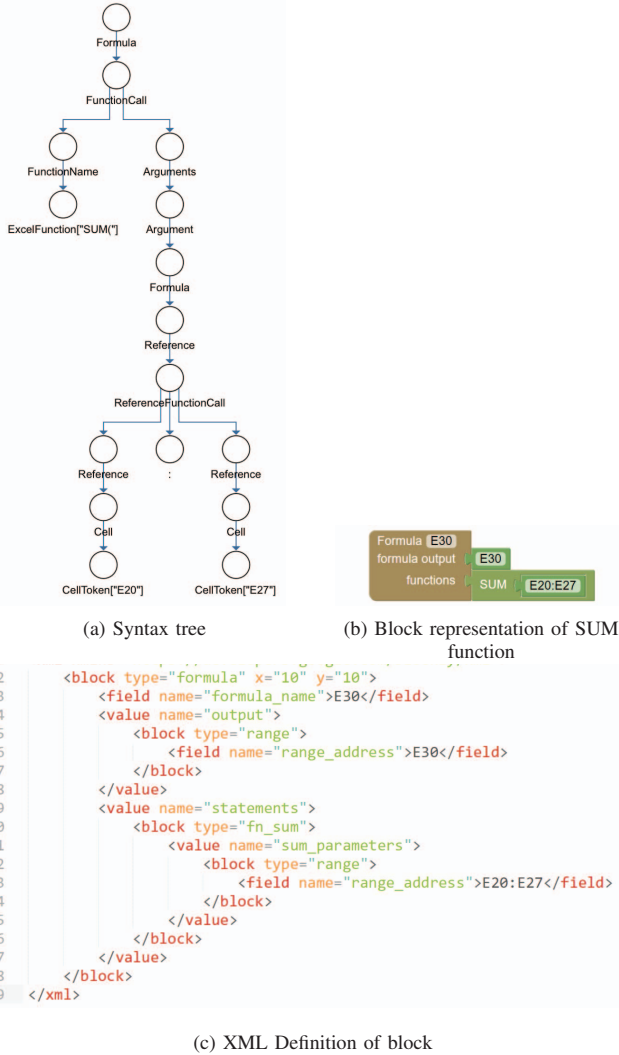
[2]https://doi.org/10.6084/m9.figshare.14268017.v1

289

(a) Syntax tree

(b) Block representation of SUM function

```
2    <block type="formula" x="10" y="10">
3        <field name="formula_name">E30</field>
4        <value name="output">
5            <block type="range">
6                <field name="range_address">E30</field>
7            </block>
8        </value>
9        <value name="statements">
0            <block type="fn_sum">
1                <value name="sum_parameters">
2                    <block type="range">
3                        <field name="range_address">E20:E27</field>
4                    </block>
5                </value>
6            </block>
7        </value>
8    </block>
9 </xml>
```

(c) XML Definition of block

Fig. 2. Generation of block-based formula

B. Generate Block-based Formulas

We extended XLBlocks with the functionality to generate a block-model of a formula from the textual formula. To do so, we use XLParser, a parser developed by Aivaloglou *et. al* [27] [28] that produces a parse tree for spreadsheet formulas. In XLBlocks, this parse tree is converted into an XML definition of the block model, which is then translated to the formula's block-based representation (see Figure 2).

C. Highlighting and scrolling

Spreadsheet formulas often use the outcome of other formulas in their calculation. To completely understand a formula, users need to trace the formula's precedents. To support users in understanding the formula and tracing its precedents, XLBlocks highlights cells in the spreadsheet that are referred to in the formula. In Figure 1 a part of the formula in XLBlocks is selected (highlighted by a yellow border). The

cells referred to in this selected part of the formula are also highlighted in the spreadsheet.

Some cells might be out of the user's field of view. To inspect these cells, a user can select a single range block, and XLBlocks will automatically scroll the cursor to that cell and make it the active cell. If needed, the user can immediately inspect that formula and navigate from precedent to precedent to analyze the complete calculation chain. For comprehension, the user must see the cell, its content, and, preferable, the corresponding label. In most spreadsheet models, the label can be found to the left or above the cell. Therefore, XLBlocks ensures that the columns to the left and the rows above the cell are visible when a user scrolls to a cell.

D. Implementation

XLBlocks has been implemented as an Excel Add-in. It has been developed with the Excel JavaScript API [29]. The Blockly Library [30] [23] is used to develop the visual programming editor and was extended with custom blocks and code generators for the definition and generation of spreadsheet formulas. Twenty-tree different spreadsheet functions have been implemented in the research prototype of XLBlocks. Among these twenty-three functions are the fifteen most frequently used functions of the Enron Corpus [31]. The current research prototype of XLBlocks can only analyze formulas on the same worksheet.

IV. Design Think-Aloud Study

A. Research Questions

Spreadsheets have a long lifespan, and thus spreadsheets need maintenance. Furthermore, there will be several transfer moments during their lifespan where knowledge about the spreadsheet needs to be exchanged between different users. For both maintenance tasks and the transfer scenarios, a thorough understanding of the formulas is essential.

Therefore we focus in this study on the effect of a block-based formula language on formula comprehension in spreadsheets. In this paper, we will answer the following research questions:
:
- *RQ1:* What is the effect of a block-based formula editor on the users' ability to understand a formula?
- *RQ2:* What is the effect of a block-based formula editor on the users' ability to explain a formula to somebody else?
- *RQ3:* What is the effect of a block-based formula editor on the users' ability to understand the spreadsheet model as a whole?

B. Participants

We invited forty-three professional Excel users by e-mail from twenty-eight different companies. We were looking for experienced Excel users that use Excel in their professional lives. Twenty-one of them, working for thirteen companies, accepted the invitation (see Table I). Five of them had participated in our earlier study [12].

290

Nr.	Gender	Age	Functional Domain	L^a	F^b	E^c
P1	M	52	Engineering	7	D	35
P2	F	48	Controlling	8	D	25
P3	M	38	Controlling	8	D	25
P4	M	44	IT	8	D	20
P5	M	59	Finance & Control	7	D	35
P6	M	38	Consultancy	9	D	25
P7	M	43	Finance	8	D	20
P8	M	60	Finance	6	D	30
P9	M	38	Finance	8	D	18
P10	M	64	Finance	8	D	30
P11	M	55	Data analytics	8	W	30
P12	F	46	Business Intelligence	8	D	20
P13	M	49	Finance & Control	7	D	25
P14	M	49	Consultancy	6	D	25
P15	F	60	Consultancy	7	D	25
P16	M	55	General Management	7	M	25
P17	M	45	Business Control	7	D	20
P18	M	44	Finance & Control	8	D	24
P19	M	48	Finance	8	D	25
P20	M	38	Finance	9	D	20
P21	M	55	Finance	8	D	29
Average		49		8		25

^{a}Excel level, ^{b}Frequency: (D)aily, (W)eekly, (M)onthly, ^{c}Experience (yrs)

All participants use Excel professionally, are accustomed to working with formulas, and have on average twenty-five years of experience with Excel. Most of them use Excel daily. We asked them to assess their level of expertise with Excel on a ten-point scale (one = low, ten is high). This form of rating is widely used in (European) schools, and the participants are more familiar with it than, for example, a five-point Likert scale. On average, they rated themselves an eight out of ten.

TABLE II
PACIONE'S COMPREHENSION ACTIVITIES

Activity	Description
A1	Investigating the functionality of the system
A2	Adding to or changing the system's functionality
A3	Investigating the internal structure of an artifact
A4	Investigating dependencies between artifacts
A5	Investigating run-time interactions in the system
A6	Investigating how much an artifact is used
A7	Investigating patterns in the system's design
A8	Assessing the quality of the system's design
A9	Understanding the domain of the system

C. Comprehension Tasks and Think-Aloud Study

In a think-aloud study, we asked participants to perform twelve comprehension tasks on an existing spreadsheet model. We used the framework designed by Pacione *et. al* [32], commonly used for empirical evaluation of code comprehension [33] [34] and used their set of nine comprehension activities (see Table II)

In previous research [35] we have translated Pacione's software comprehension tasks to the spreadsheet domain. In this study, we use similar tasks. Each tasks covers at least one of the activities in Table II and all tasks combined to cover all activities (see Table III).

Because of the liveness of a spreadsheet, there is no clear distinction between coding and runtime. We, therefore, excluded activities A5 and A7.

The spreadsheet we use for the study is defined by the Dutch Primary Education Board. Schools can use it to calculate the annual salary costs of their employees. We choose this model because it is publicly available[3], and contains twelve of the fifteen most frequently used functions in the Enron Corpus [31]. We adapted the model to incorporate the three missing functions and moved all lookup tables to the same sheet as the calculation model[4].

D. Think-Aloud study

We organized on-line meetings with the twenty-one participants. We used either Microsoft Teams or GoToMeeting to facilitate these meetings. At the start of every meeting, we checked the participants' monitor and resolution, shared our screen with them, and asked if the different interface elements were readable. For one participant, we changed the zoom factor of the XLBlocks interface from 80% to 110%.

Before the meeting, we sent each participant an instruction video about XLBlocks, asking them to watch it before participating in the study. Two participants were not able to do this. At the start of their meeting, we provided the instruction live, using the same script that was used for the video.

We shared our screen with the participants during the meeting and gave them control over our keyboard and mouse. We also recorded the meeting on video. We asked the participants to perform the twelve comprehension tasks and to think-aloud during the study. If they felt silent while performing the tasks, we gave them a quick reminder to express their thoughts.

E. Interview

Immediately after the comprehension tasks, we conducted a 45-minute interview. We used the Cognitive Dimensions of Notation (CDN) framework to structure the interview. The CDN Framework has been used in several usability studies [15], [36]–[39], and Blackwell and Green developed a questionnaire for it [40]. In our interview, we covered the dimensions as defined in [13]. We excluded the dimension *Abstract Gradient* because, in XLBlocks, users can not create their own blocks.

We did not ask the participants to fill out the CDN questionnaire, but rather, we used the questions to structure our interview[5]. It allowed us to clarify a question, and it enabled us to probe participants for additional details.

We grouped all participants' answers per cognitive dimension and complemented them with our observations and the participant's remarks from the think-aloud study. We used the combined information to answer our research questions. These findings will be presented in the next section.

[3]https://www.poraad.nl/files/themas/financien/werkgeverslasten_po_2020.xlsx
[4]Adapted model available at: https://doi.org/10.6084/m9.figshare.14268017.v1
[5]Interview questions available at:https://doi.org/10.6084/m9.figshare.14268017.v1

TABLE III
OVERVIEW OF COMPREHENSION TASKS

Nr.	Task	Comprehension Activities							Total
		A1	A2	A3	A4	A6	A8	A9	
T01	Explain a calculation	X		X				X	3
T02	Adapt a calculation		X	X					2
T03	Explain a key concept of the model	X		X				X	3
T04	Find and correct an error		X	X					2
T05	Correct an error		X	X					2
T06	Determine relationships between two cells				X	X			2
T07	Find dependents of a cell				X	X			2
T08	Explain how the spreadsheet can be improved			X			X		2
T09	Assess adaptability of the spreadsheet						X		1
T10	Assess transferability of the spreadsheet						X		1
T11	Explain a calculation	X		X				X	3
T12	Explain a calculation	X		X				X	3
Total		4	3	8	2	2	3	4	26

V. RESULTS

In this section, we present the results of the think-aloud study. First, we will cover the execution of the different comprehension tasks, after that the findings from the CDN interview, and we end this section by answering the research questions.

A. Comprehension Tasks

All twenty-one participants were able to perform ten of the twelve comprehension tasks. In the next paragraphs, we will describe our findings in detail. We grouped the findings by the comprehension activities as defined by Pacione *et al.* [32].

1) Investigating the functionality of (a part of) the system: By design, XLBlocks displays a single formula. Nevertheless, several features helped the participants to get an understanding of the spreadsheet model as a whole.
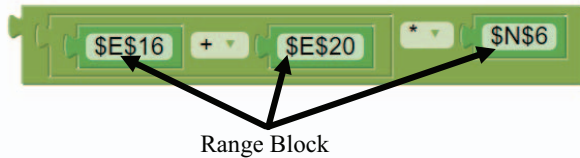

Fig. 3. Example of range blocks in XLBlocks

In the block-based representation of the formula, each cell reference is represented by a range block (see Figure 3). If a user clicks on any block in the formula, XLBlocks will highlight all cells in the spreadsheet with a range block in that part of the formula. This gives the user an overview of the other cells in the spreadsheet that are involved in the calculation of this formula. If users click on a single range block, they can open that formula in XLBlocks to analyze it. In this way, users are supported in understanding the working of the spreadsheet model as a whole.

In the current implementation, XLBlocks highlights all cells in the same color. Some participants suggested that it would be helpful if the highlighted cells had a unique color and that the selected range blocks would light up in the same color.

2) Adding or changing the system's functionality: We confronted the participants with two erroneous formulas, and all participants were able to find and correct the errors in these formulas. Furthermore, they were able to make a change in a complex formula. They had to replace a nested IF structure with a VLOOKUP function. According to the participants, this was easier to perform in XLBlocks than in the formula bar. In XLBlocks, it is possible to drag the complete IF structure out of the formula and replace it with a VLOOKUP function block. They did not need to consider the exact placement of parentheses and commas', making the change more straightforward and quicker.

One of the errors the participants needed to find was a SUM function in which some cells were omitted. The highlighting of the involved cells in the spreadsheet helped to visualize the mistake, but Excel offers the same functionality in the formula bar. Eight participants remarked that for such a small change (extending the range of a SUM function), the formula bar is more efficient than XLBlocks.

The second error that participants had to correct was a logic error in an IF formula. In this case, according to the participants, it was easier to spot the error in XLBlocks because the formula as a whole was easier to read.

3) Investigate the internal structure of an artifact: We asked the participants to explain several formulas with XLBlocks. All participants were able to do that, even if they had no domain knowledge of the spreadsheet model. In XLBlocks, in contrast to the formula bar, the parameters of a function are named. According to the participants, that makes it easier to understand a formula. For example, P19 stated: *"In XLBlocks, the IF, THEN, and ELSE parts of the formula are visible in the IF Block. That is not the case in the Excel formula bar."*

When participants explained a formula, we observed that it took significant time to look up a cell reference in the spreadsheet to determine its meaning. If the same reference is used more than once in the formula, it was not uncommon that participants had to look up the meaning again. Some participants suggested that formulas would be easier to read if the cell references had meaningful names instead of the abstract A1 naming style that is the default in Excel (see also

292

Section VI-B).

Several participants noted that in XLBlocks, a formula is visually split into different components (instead of a long string of characters in the formula bar), making it easier to understand the formula.

4) Investigating dependencies between artifacts and how much an artifact is used: The participants were asked to trace the precedents of five different formulas. They were all able to do that. XLBlocks provide two features that supported them in these tasks:

1) When the formula is selected in XLBlocks, all cell references used in the formula are highlighted in the spreadsheet.
2) If a user selects a single cell reference in XLBlocks, they can immediately inspect the formula in that cell and quickly navigate from formula to formula.

5) Assessing the quality of the system's design: We asked the participants how they would improve the spreadsheet model we used during the comprehension tasks. Two of the twenty-one participants were not able to come up with some improvements. The other participants mentioned improvements like:

- separating input, calculation, and output of the model to different worksheets
- move all parameters and lookup tables to different worksheets
- in the current model, some calculation are based on monthly amounts, other on yearly amounts. Several participants proposed to make these calculations consistent. Either all based on monthly or on yearly amounts, but not mixing them.

6) Understanding the domain of the system: After working with the spreadsheet model in XLBlocks, participants had a better understanding of the functional domain (salary administration). One of the questions was if they could tell which components make up the total salary costs. Furthermore, they had to explain the calculation of several pension premiums. To calculate this, you have to take into account an exemption amount. If your income is lower than the exemption amount, you do not have to pay a premium; however, if your income is higher than the exemption amount, you pay a percentage of your income minus the exemption amount. When the participants had to explain these types of formulas, they first read the formula aloud, and next, they would summarize the logic of the formula in their own words, recognizing the pattern described above.

B. CDN Interview

In the next paragraphs, we will present the results of the interview. We will group the observations by the different Cognitive Dimensions [13].

1) Viscosity: The dimension *Viscosity* expresses the resistance to change in a language and consequently has more impact on maintenance than comprehension. We included it in the interview because one of the comprehension activities

(A2 in Table II) involves adding or changing the system's functionality. XLBlocks has a drag and drop interface. The mouse or trackpad is the primary input device. In general, this slowed the user down, which several participants confirmed. Because of this, participants would prefer to make small changes in the formula bar instead of XLBlocks.

For complex formulas, this was different. In one of the tasks, they had to replace a nested IF structure with a VLOOKUP function. This is not easy in the formula bar and involves careful placement of the cursor, ensuring that one is not selecting one parenthesis too many or too few. Implementing this change goes faster in XLBlocks. The user does not have to bother about parentheses and can easily drag the IF structure out of the formula and replace it with a VLOOKUP block.

Finally, some participants pointed out that it can be challenging to click on a range block, mostly when used in an inline function block (for example, range E16 in Figure 3). If the click is not precisely on the range block, the function block is selected instead.

2) Visibility: All participants were able to see the complete formulas at a glance. To understand the formula, it is also essential to see which cells are referenced, which can be problematic in a large spreadsheet. If users inspect a formula in the formula bar, they have to scroll to the cells they can not see. If a user selects a range block in XLBlocks, it will automatically scroll to the corresponding cell in the spreadsheet.

Several participants commented that multiple nested binary operations (see, for example, the else clause in Figure 4b) were difficult to understand. Function blocks have different colors, depending on their category. All binary functions have the same category (Math and Trigonometry) and, therefore, the same color. Furthermore, each function block has its own border, but it is thin and subtly colored. These issues combined make it difficult to distinguish the individual functions.

3) Premature Commitment: In XLBlocks, the user can build a formula in any order. Also, it is possible to change the order of the functions within the formula at any given time. The only requirement is that the output of the top-level function is connected to a formula block.

Nevertheless, some participants had the feeling they had to start with the formula block and build the formula from there, starting with the top-level function. They even said they liked how the blocks would force them to build the formula in a structured manner. For example, P1 said: *"You are somewhat forced in a structure, and I actually like that."*, and P7: *"I think, because it is visual, you are forced to think about the formula you are building."*

Other participants recognized that they could start anywhere in the formula. All participants agreed that it is easy to change the order of functions in the formula.

4) Hidden Dependencies: In a spreadsheet, there are dependencies between functions in a formula and between cells in the spreadsheet. Participants indicated that it is easy to see the dependencies between functions in a formula. Each function

293

is visualized as a puzzle piece, and the connection between two pieces visualizes the dependency between the functions.

Concerning the spreadsheet level, participants liked that XLBlocks would highlight all cells in the spreadsheet used in the formula. Also, the possibility to automatically scroll to a precedent cell in the spreadsheet by clicking on the corresponding range block was appreciated. P6 said: *"Clicking on highlighted cells and jumping from one formula to another makes it easier to explain the spreadsheet"*, or according to P7: *"You can click on a cell and immediately jump to that formula, that for me is one of the biggest advantages."* Furthermore, they liked the option that once a precedent cell was selected in XLBlocks, they could inspect that cell's formula in XLBlocks.

Unfortunately, once they jumped to one of the precedent cells, it was impossible to jump back to the original formula. They also indicated that it was not possible in XLBlocks to see depending cells of a formula. When they had to trace dependents during the comprehension tasks, they had to fall back to Excel's native trace dependents function.

5) Role-Expressiveness: Participants said that in XLBlocks, they could 'see' the formula. They named the IF function as an example. *"I think you will make fewer logic errors because you really see the formula"* (P6), *"The structure is clear, it is more transparent, you see it immediately."* (P11), and *"It is very easy to read the IF, THEN, ELSE formula."* (P21). In the formula bar (see Figure 4a), the IF function is displayed as a single string. A comma separates the THEN and ELSE parts, and by convention, the second argument is the THEN part, and the third argument the ELSE part. In XLBlocks (see Figure 4b), the IF, THEN, and ELSE parts are labeled and visualized on three different lines. Furthermore, each block has a thin border that acts as the equivalent of parentheses. According to the participants, these features combined made it easier to understand the formula.

(a) The IF function as a long string in the formula bar
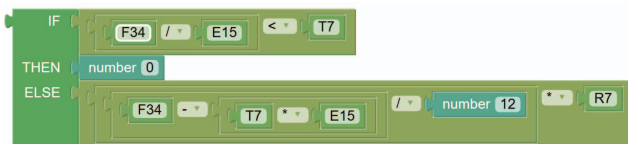
(b) The IF function split over several lines with named parameters in XLBlocks

Fig. 4. Two variant of the IF Function

XLBlocks, like the formula bar, does not provide an explanation about the function parameters. When working with the VLOOKUP function, participants indicated that they were unsure about the meaning of some of the parameters, and explanation would have helped. This could easily be solved in XLBlocks by adding tool-tips to the parameters.

6) Error-Proneness: As was already mentioned at *Premature Commitment*, some participants pointed out that the blocks guide you through a function's structure. It is not possible to forget a parameter and, because they are labeled, one can not confuse them. As a consequence, this leads to fewer errors. Additionally, most participants noted that XLBlocks places the parentheses and commas automatically, which further reduces possible errors.

Some participants remarked that the operators in the binary function blocks (see Figure 7b) are hard to read, and it is easy to forget to change the default operator. Both cases would lead to an incorrect formula.

7) Secondary Notation: XLBlocks has a dedicated comment block (see Figure 5) that can be used to annotate a formula. Multiple comment blocks can be added to a formula, and they can be placed freely on the canvas.
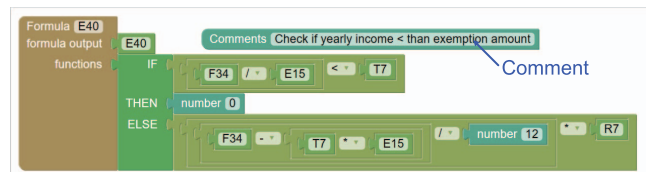

Fig. 5. Example of a comment block

Participants said that they would use the comment block to document the formula, describe its purpose, explain the calculation, and describe the meaning of the cells used in the calculation. The current comment block has been designed to accommodate short comments, but several participants indicated they would like to enter larger text blocks.

8) Closeness of Mapping: Cells used in a formula are visualized in XLBlocks with range blocks (see Figure 3). The cells are identified by their cell address in A1 notation. When participants had to explain formulas, they had to look up the cell by their address in the spreadsheet to see its functional meaning. While explaining a complex formula, it occurred more than once that they forgot the meaning of a cell and had to look it up again, which would slow down the explanation. Several participants mentioned they would prefer the possibility to give the range blocks a meaningful name. An example can be found in Figure 6. Figure 6a shows the formula as it is currently visualized in XLBlocks, and in Figure 6b the cell addresses are replaced by meaningful names.
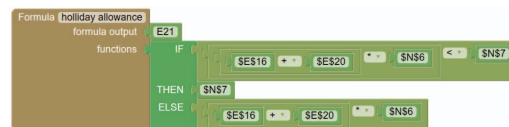
(a) Cell references in A1 style

(b) Cell references with meaningful names

Fig. 6. Different styles for cell references

294

9) Consistency: Participants recognized that blocks with the same purpose have the same color. Constants have a different color than cell references, and Lookup functions have a different color than logical functions. The use of color helped them to understand the formulas better. Some participants remarked that it would be even better if the colors would be repeated in the toolbox menu. In that case, it is easier to derive the meaning of a specific color.

There are over 450 functions in Excel. It is not feasible to give every function a unique color. For that reason, functions of the same type (as defined by Microsoft[6], such as math and trigonometry, logical, and lookup and reference) have the same color. If in a formula several functions of the same type are nested (for example, the addition, multiplication, and comparison in the IF clause of Figure 6a), this will lead to a group of blocks with the same color. According to the participants, this makes it less easy to read and interpreter the formula. Some participants argued that maybe the color of a function block should depend on the formula's level of nestedness.

If a user selects (a part) of a formula in XLBlocks, the used cells are highlighted in the spreadsheet. These cells get all the same highlight color. Some participants suggested giving each highlighted cell its own color and using that color to highlight the formula's corresponding range block.

10) Diffuseness: According to the participants, the size of the formulas is adequate. This is remarkable because block-based languages tend to be more diffuse [41] and programmers value that as a negative because less code will fit on the screen. However, spreadsheet users are accustomed to entering their formulas in a tiny formula bar. They are relieved that in XLBlocks, they have more space available, and because XLBlocks focuses on one formula at a time, even complex formulas will fit on the canvas and do not require additional scrolling.
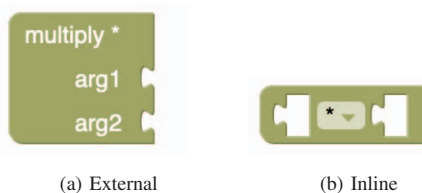

(a) External   (b) Inline

Fig. 7. Different input types

XLBlocks handles the input of a function block in two different ways: external (Figure 7a) and inline (Figure 7b). Basic functions such as addition, subtraction, and division use the inline variant. The inline variant is more natural to read, but if several of these functions are concatenated, one ends up with a wide formula block (see, for example, the else clause in Figure 4b). The participants confirmed that several binary operations after each other take more space than needed. According to the participants, also the constant blocks

---

6https://support.microsoft.com/en-us/office/excel-functions-alphabetical-b3944572-255d-4efb-bb96-c6d90033e188

(number, text, and boolean) are relatively large in relation to their importance in the formula (see, for example, the number block in Figure 4b).

11) Hard Mental Operations: According to the participants, working with XLBlocks does not require more mental effort than working with formulas in the formula bar. They provide two reasons for this:

- Because of the visualization in blocks, the formula is split into smaller components. This makes it easier to understand the formula. Participants do not have to understand the formula at once but can focus on a smaller component. For example, P1 noted: *"I do not have to split the formula into smaller parts, XLBlocks does that for me, which makes it easier to understand."*
- In XLBlocks, the user does not have to think about the placement of parentheses, quotes, or commas.

12) Provisionality: Participants loved the fact that they could play with formulas in XLBlocks. P13: *"Dragging a part of formula out of the formula is very easy"* and P19: *"It is very visual, you can drag a part out of it and paste it back in very easily, I really like that."*

They could easily drag components out of the formula onto the canvas and replace them with other functions or components. A component that is dragged out of a formula can be parked and saved on the canvas. It will not influence the generation of the spreadsheet formula. If they were not satisfied with their changes, they could quickly revert to the previous state. It gave them also the opportunity to evaluate two or more variations of the same formula. Finally, they liked the opportunity to change the order of functions within a formula easily.

13) Progressive Evaluation: Participants indicated that it is possible to stop working on a formula at any time. The formula does not need to be correct before it can be saved, which is not the case in Excel's formula bar. It is possible to test a part of a formula as long as it will lead to a valid formula expression. Seeing if a formula was finished was easy, according to the participants. As long as there are no missing puzzle pieces, the formula is finished. In Excel's formula bar, it is much harder to see if an argument of a function, a parenthesis, or comma is missing.

Fig. 8. Formula wizard in Excel displaying intermediate results

The participants indicated that they were missing intermediate results in XLBlocks. This is possible in the formula wizard in Excel (see Figure 8), and they would like to see similar functionality in XLBlocks.

295

C. Research Questions

In the final paragraphs of this section, we will answer the research questions.

1) RQ1 understand a formula: Participants believe that, for complex formulas, it is easier to understand them with XLBlocks than with the formula bar. They have several arguments for this. First, the formula's block-based representation splits the formula into smaller components, making it easier to comprehend. Secondly, the parameters of a function are named. Less knowledge of the function syntax is needed to understand it, and lastly, if the user clicks on the formula in XLBlocks, all cells used in the formula are highlighted in the spreadsheet. This enables them to see which numbers are used in the calculation.

2) RQ2 explain a formula to somebody else: According to the participants, XLBlocks supports the user in explaining the formula. When they click on a part of a formula, XLBlocks highlight the blocks in that part of the formula. This helps in focusing the explanation on a specific part of the formula. Furthermore, comment blocks can be used to document the purpose of a formula and explain components in the formula, such as explaining the test performed in an IF statement. Finally, by clicking on the individual range blocks in the formula, participants could navigate the spreadsheet, highlighting the cells used in the formula and explaining the numbers' functional meaning in these cells.

3) RQ3 understand the spreadsheet model as a whole: XLblocks focuses on a single formula at a time. Nevertheless, participants were able to get an understanding of the working of the spreadsheet as a whole. Even if they were not familiar with the functional domain of the spreadsheet (payroll administration). Participants could easily click from one formula to another to see how the different formulas were related to each other and form an opinion about the workings of the model. Also, the possibility of automatically scrolling to the different cells and quickly reading the labels helped to understand the functional meaning of the calculation.

VI. DISCUSSION

A. Confusing IF statement

During the think-aloud study, several participants got confused when explaining a formula that contained an IF function (see Figure 9). In this formula, the outcome of an IF function is multiplied with the addition of two percentages. Because of how the blocks are visualized, the multiplication of the sum of two percentages is displayed at the same height as the IF statement. This led the participants to believe that the multiplication was a part of the IF statement, leading to a logical test that did not make any sense when translated into business terms.

Several factors are causing this misconception.

- Both the operand and the arguments of the binary function are aligned at the top. If the multiplication symbol and the addition of the two percentages had been aligned
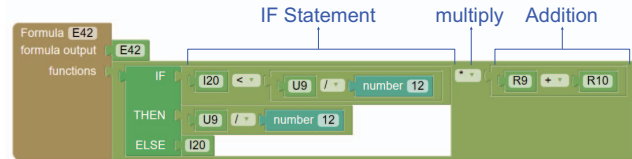

Fig. 9. Several participants struggled to explain this formula

at the middle of the block, the formula would be less confusing.

- Except for the IF function itself, all other functions in the formula are binary functions and have the same color, making it difficult to distinguish them from each other.
- Each function has its own border, but it is very thin with a light gray color to simulate a 3D effect. This makes it challenging to see where one function ends, and another begins.

Aligning the operands and arguments at the middle of a block and making the borders of function a fraction thicker with a more contrasting color would prevent this kind of misconception.

B. Giving Range Blocks Meaningful Names

As described in the previous section, several participants suggested that the readability of formulas would benefit if range blocks got meaningful names instead of an abstract cell address (see also Figure 6b). In Excel, it is possible to assign names to ranges, and several authors have advocated the use of range names [42], [43]. However, other studies indicate that there are inherent risks in using named ranges. Panko and Ordway [44] identify the risk that named ranges can appear correct in the formula but refer to the wrong range, and McKeever [45], [46] found that the use of named ranges decreases the ability of novice spreadsheet users to find and correct errors. The question, if it would be possible to implement range names in XLBlocks in such a way that it would not hinder the debugging process and makes it easy to see to which range the name refers, would make for interesting future work.

C. Navigating Formulas

While performing task *T06 Determine relationships between two cells*, participants noted that it is easy to navigate in XLBlocks to a direct precedent of the current formula but not to navigate back. This could be solved in XLBlocks by displaying a breadcrumb trail at the top of XLBlocks' canvas. It would show a horizontal list of formulas that the user has analyzed, and by clicking on any of the formulas in the list, the user would navigate back to that formula. Another solution that could be implemented complementary to the breadcrumb trail would be a browser-like navigate back button.

D. Intermediate Results

When answering questions about the cognitive dimension *Progressive Evaluation* (see Section V-B), participants indi-

296

cated that it would be even easier to evaluate formulas if, in XLBlocks, they could see the intermediate results of the calculation. Inspired by the work of Leber *et. al* [47], Figure 10 shows an example of how this could be implemented in XLBlocks.
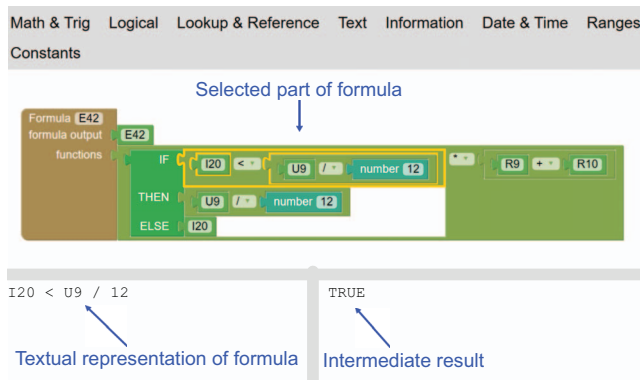

Fig. 10. Showing textual formula and intermediate results in XLBlocks

If the user selects a part of a formula, a textual representation of that part of the formula is displayed at the bottom left of the XLBlocks interface, while on the bottom right, the result of the calculation is displayed. One could even consider making the textual representation of the formula also editable. We keep this as a point for future work.

E. Threats to Validity

A threat to the external validity of our think-aloud study concerns the representativeness of the participants. Additional studies are necessary to generalize our findings.

Furthermore, there is a risk of aptitude treatment interaction since participants of a previous study were also invited to participate in this study. It could be the case that only the most positive ones responded to this request. Eventually, only five of the twenty-one participants were also a participant in the previous study, and judging by the number of points of criticism we received from them during the think-aloud study and the interview, we believe our finding were not impacted by the aptitude treatment.

Another threat to the external validity is the representativeness of the comprehension tasks. We mitigated this by using a validated set of comprehension tasks defined by Pacione *et. al* [32].

Participants were selected from our network, which is a threat to the internal validity of our study. However, we believe the current group serves as a useful reference group, as the persons were experienced professional spreadsheet users, came from different companies, and worked in different functional domains.

We fulfill the role of both developer of XLBlocks and an interviewer during the think-aloud study. This is a threat to the internal validity of the study. We lessened this risk by using a validated set of comprehension tasks and using the CDN Framework to guide our questions during the interview,

ensuring that all aspects of the usability of the XLBlocks interface were covered.

VII. CONCLUDING REMARKS

The purpose of this paper is to research the effect of a block-based language on formula comprehension in spreadsheets. We extended the block-based formula editor XLBlocks with functionality to generate block-based representations of existing spreadsheet formulas. We asked participants in a think-aloud study to perform twelve comprehension task and immediately after they finished these tasks, interviewed them about their experience with XLBlocks.

Participants believed that XLBlocks helped them to understand formulas better. They argued that the formula's visualization in blocks helped separate smaller parts in the formula, making it easier to comprehend. Furthermore, in XLBlocks, each function argument had a descriptive label, making the formula more comfortable to read. Finally, the formula's cells were also highlighted in the spreadsheet, making it easier to see what was calculated by the formula.

During the study, participants had to explain three different formulas using XLBlocks. According to participants, XLBlocks made it easier to do so. They could select a part of a formula during an explanation, and XLBlocks would highlight the relevant blocks. This helped both the participants and the listener to focus their attention on this part of the formula. Participants also noted that it is possible to document a formula's workings with the dedicated comment blocks of XLBlocks. Finally, if a formula was part of a larger calculation chain, users could easily navigate the formula's precedents to show how the formulas worked together.

This mechanic of navigating between formulas was, according to the participants, also instrumental in gaining an understanding of the spreadsheet's workings. The possibility of quickly navigating to cells in the spreadsheets by selecting a cell reference in XLBlocks and looking up the corresponding labels in the spreadsheet helped to gain more insight into the spreadsheet's problem domain.

This research gives rise to several directions for future work. Tracing relations between formulas is instrumental in understanding the workings of a spreadsheet. We have seen that XLBlocks can help in finding precedents of a formula but lacks functionality in tracing dependents. We will investigate what would be the best way to extend XLBlocks with this functionality. Furthermore, we will extend XLBlocks with the possibility to show the textual representation of (a part of) the formula and the corresponding intermediate results of the calculation in real-time. It will enable users to receive direct feedback on changes they make in the formula. Finally, we will explore the possibility of giving cell references in XLBlocks a meaningful name.

VIII. DATA AVAILABILITY

A video of XLBlocks, the source code of XLBlocks, the spreadsheet model used during the Think-Aloud study, and the interview questions used are available on figshare, DOI 10.6084/m9.figshare.14268017

REFERENCES

[1] R. R. Panko, "Spreadsheet errors: What we know. what we think we can do," *arXiv preprint arXiv:0802.3457*, 2008.

[2] K. J. Rothermel, C. R. Cook, M. Burnett, J. Schonfeld, T. R. Green, and G. Rothermel, "Wysiwyt testing in the spreadsheet paradigm: An empirical evaluation," in *Software Engineering, 2000. Proceedings of the 2000 International Conference on*. IEEE, 2000, pp. 230–239.

[3] S. Roy, F. Hermans, and A. van Deursen, "Spreadsheet testing in practice," in *Software Analysis, Evolution and Reengineering (SANER), 2017 IEEE 24th International Conference on*. IEEE, 2017, pp. 338–348.

[4] F. Hermans, M. Pinzger, and A. Deursen, *ECOOP 2010 – Object-Oriented Programming: 24th European Conference, Maribor, Slovenia, June 21-25, 2010. Proceedings*. Berlin, Heidelberg: Springer Berlin Heidelberg, 2010, ch. Automatically Extracting Class Diagrams from Spreadsheets, pp. 52–75. [Online]. Available: http://dx.doi.org/10.1007/978-3-642-14107-2_4

[5] J. Cunha, M. Erwig, and J. Saraiva, "Automatically inferring classsheet models from spreadsheets," in *Visual Languages and Human-Centric Computing (VL/HCC), 2010 IEEE Symposium on*. IEEE, 2010, pp. 93–100.

[6] F. Hermans, M. Pinzger, and A. van Deursen, "Detecting and refactoring code smells in spreadsheet formulas," *Empirical Software Engineering*, pp. 1–27, 2014.

[7] J. Cunha, J. P. Fernandes, H. Ribeiro, and J. Saraiva, "Towards a catalog of spreadsheet smells," in *Computational Science and Its Applications–ICCSA 2012*. Springer, 2012, pp. 202–216.

[8] D. W. Barowy, D. Gochev, and E. D. Berger, "Checkcell: Data debugging for spreadsheets," in *ACM SIGPLAN Notices*, vol. 49, no. 10. ACM, 2014, pp. 507–523.

[9] S. Badame and D. Dig, "Refactoring meets spreadsheet formulas," in *Software Maintenance (ICSM), 2012 28th IEEE International Conference on*. IEEE, 2012, pp. 399–409.

[10] F. Hermans and D. Dig, "Bumblebee: a refactoring environment for spreadsheet formulas," in *Proceedings of the 22nd ACM SIGSOFT International Symposium on Foundations of Software Engineering*. ACM, 2014, pp. 747–750.

[11] F. Hermans, M. Pinzger, and A. van Deursen, "Supporting professional spreadsheet users by generating leveled dataflow diagrams," in *Proceedings of the 33rd International Conference on Software Engineering*. ACM, 2011, pp. 451–460.

[12] B. Jansen and F. Hermans, "Xlblocks: a block-based formula editor for spreadsheet formulas," in *2019 IEEE Symposium on Visual Languages and Human-Centric Computing (VL/HCC)*. IEEE, 2019, pp. 55–63.

[13] A. F. Blackwell, C. Britton, A. Cox, T. R. Green, C. Gurr, G. Kadoda, M. Kutar, M. Loomes, C. L. Nehaniv, M. Petre *et al.*, "Cognitive dimensions of notations: Design tools for cognitive technology," in *International Conference on Cognitive Technology*. Springer, 2001, pp. 325–341.

[14] M. M. Burnett, J. W. Atwood, R. W. Djang, J. Reichwein, H. J. 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.

[15] R. Abraham, M. Erwig, S. Kollmansberger, and E. Seifert, "Visual specifications of correct spreadsheets," in *Visual Languages and Human-Centric Computing, 2005 IEEE Symposium on*. IEEE, 2005, pp. 189–196.

[16] G. Engels and M. Erwig, "Classsheets: automatic generation of spreadsheet applications from object-oriented specifications," in *Proceedings of the 20th IEEE/ACM international Conference on Automated software engineering*. ACM, 2005, pp. 124–133.

[17] J. Cunha, J. Mendes, J. Saraiva, and J. P. Fernandes, "Embedding and evolution of spreadsheet models in spreadsheet systems," in *2011 IEEE Symposium on Visual Languages and Human-Centric Computing (VL/HCC)*, Sep. 2011, pp. 179–186.

[18] R. Leitão and C. Roast, "Developing visualisations for spreadsheet formulae: towards increasing the accessibility of science, technology, engineering and maths subjects," in *9th Workshop on Mathematical User Interfaces*, 2014.

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

[20] E. P. Glinert, "Towards second generation interactive graphical programming environments," in *Proceedings of IEEE Workshop onVisual Language. IEEE CS Press, Silver Spring, MD*, 1986, pp. 61–70.

[21] M. Conway, R. Pausch, R. Gossweiler, and T. Burnette, "Alice: a rapid prototyping system for building virtual environments," in *CHI Conference Companion*. Citeseer, 1994, pp. 295–296.

[22] M. Resnick, J. Maloney, A. Monroy-Hernández, N. Rusk, E. Eastmond, K. Brennan, A. Millner, E. Rosenbaum, J. Silver, B. Silverman, and Y. Kafai, "Scratch: Programming for all," *Commun. ACM*, vol. 52, no. 11, pp. 60–67, Nov. 2009. [Online]. Available: http://doi.acm.org/10.1145/1592761.1592779

[23] N. Fraser, "Ten things we've learned from blockly," in *2015 IEEE Blocks and Beyond Workshop (Blocks and Beyond)*, Oct 2015, pp. 49–50.

[24] D. Weintrop, A. Afzal, J. Salac, P. Francis, B. Li, D. C. Shepherd, and D. Franklin, "Evaluating coblox: A comparative study of robotics programming environments for adult novices," in *Proceedings of the 2018 CHI Conference on Human Factors in Computing Systems*, ser. CHI '18. New York, NY, USA: ACM, 2018, pp. 366:1–366:12. [Online]. Available: http://doi.acm.org/10.1145/3173574.3173940

[25] D. Weintrop, H. Killen, T. Munzar, and B. Franke, "Block-based comprehension: Exploring and explaining student outcomes from a read-only block-based exam," in *Proceedings of the 50th ACM Technical Symposium on Computer Science Education*, ser. SIGCSE '19. New York, NY, USA: Association for Computing Machinery, 2019, pp. 1218–1224. [Online]. Available: https://doi.org/10.1145/3287324.3287348

[26] R. Holwerda and F. Hermans, "Towards blocks-based prototyping of web applications," in *2017 IEEE Blocks and Beyond Workshop (B B)*, Oct 2017, pp. 41–44.

[27] E. Aivaloglou, D. Hoepelman, and F. Hermans, "Parsing excel formulas: A grammar and its application on 4 large datasets," *Journal of Software: Evolution and Process*, vol. 29, no. 12, p. e1895, 2017, e1895 smr.1895. [Online]. Available: https://onlinelibrary.wiley.com/doi/abs/10.1002/smr.1895

[28] XLParser web demo. Accessed: 2021-01-29. [Online]. Available: https://xlparser.perfectxl.nl/demo/

[29] Javascript API for Office. Accessed: 2021-01-29. [Online]. Available: https://docs.microsoft.com/en-us/office/dev/add-ins/reference/javascript-api-for-office

[30] Blockly. Accessed: 2021-01-29. [Online]. Available: https://developers.google.com/blockly

[31] F. Hermans and E. Murphy-Hill, "Enron's spreadsheets and related emails: A dataset and analysis," in *Proceedings of the 37th International Conference on Software Engineering-Volume 2*. IEEE Press, 2015, pp. 7–16.

[32] M. J. Pacione, M. Roper, and M. Wood, "A novel software visualisation model to support software comprehension," in *Reverse Engineering, 2004. Proceedings. 11th Working Conference on*. IEEE, 2004, pp. 70–79.

[33] F. Hermans and E. Aivaloglou, "Do code smells hamper novice programming? a controlled experiment on scratch programs," in *Program Comprehension (ICPC), 2016 IEEE 24th International Conference on*. IEEE, 2016, pp. 1–10.

[34] B. Cornelissen, A. Zaidman, and A. Van Deursen, "A controlled experiment for program comprehension through trace visualization," *Software Engineering, IEEE Transactions on*, vol. 37, no. 3, pp. 341–355, 2011.

[35] B. Jansen and F. Hermans, "The effect of delocalized plans on spreadsheet comprehension: a controlled experiment," in *Proceedings of the 25th International Conference on Program Comprehension*. IEEE Press, 2017, pp. 286–296.

[36] M. Kauhanen and R. Biddle, "Cognitive dimensions of a game scripting tool," in *Proceedings of the 2007 conference on Future Play*. ACM, 2007, pp. 97–104.

[37] M. Bellingham, S. Holland, and P. Mulholland, "A cognitive dimensions analysis of interaction design for algorithmic composition software," 2014.

[38] F. Turbak, D. Wolber, and P. Medlock-Walton, "The design of naming features in app inventor 2," in *2014 IEEE Symposium on Visual Languages and Human-Centric Computing (VL/HCC)*. IEEE, 2014, pp. 129–132.

[39] R. Holwerda and F. Hermans, "A usability analysis of blocks-based programming editors using cognitive dimensions," in *2018 IEEE Symposium on Visual Languages and Human-Centric Computing (VL/HCC)*, Oct 2018, pp. 217–225.

[40] A. F. Blackwell and T. R. Green, "A cognitive dimensions questionnaire optimised for users." in *PPIG*, vol. 13, 2000.

[41] 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.

[42] S. Kruck, "Testing spreadsheet accuracy theory," *Information and Software Technology*, vol. 48, no. 3, pp. 204–213, 2006.

[43] P. L. Bewig, "How do you know your spreadsheet is right?" *arXiv preprint arXiv:1301.5878*, 2013.

[44] R. R. Panko and N. Ordway, "Sarbanes-oxley: What about all the spreadsheets?" *arXiv preprint arXiv:0804.0797*, 2008.

[45] R. McKeever, K. McDaid, and B. Bishop, "An exploratory analysis of the impact of named ranges on the debugging performance of novice users," 2009.

[46] R. McKeever and K. McDaid, "How do range names hinder novice spreadsheet debugging performance?" 2010.

[47] Ž. Leber, M. Črepinek, and T. Kosar, "Simultaneous multiple representation editing environment for primary school education," in *2019 IEEE Symposium on Visual Languages and Human-Centric Computing (VL/HCC)*, 2019, pp. 175–179.