Explaining Spreadsheets with Spreadsheets (Short Paper)*

Jácome Cunha
University of Minho & NOVA LINCS, Portugal
jacome@di.uminho.pt

Mihai Dan
Oregon State University, USA
danm@oregonstate.edu

Martin Erwig
Oregon State University, USA
erwig@oregonstate.edu

Danila Fedorin
Oregon State University, USA
fedorind@oregonstate.edu

Alex Grejuc
Oregon State University, USA
grejuca@oregonstate.edu

Abstract

Based on the concept of explanation sheets, we present an approach to make spreadsheets easier to understand and thus easier to use and maintain. We identify the notion of explanation soundness and show that explanation sheets which conform to simple rules of formula coverage provide sound explanations. We also present a practical evaluation of explanation sheets based on samples drawn from widely used spreadsheet corpora and based on a small user study.

In addition to supporting spreadsheet understanding and maintenance, our work on explanation sheets has also uncovered several general principles of explanation languages that can help guide the design of explanations for other programming and domain-specific languages.

CCS Concepts • **Software and its engineering** → *Domain specific languages*; *Software maintenance tools*;

Keywords Software Understanding, Explanation Principles

ACM Reference Format:
Jácome Cunha, Mihai Dan, Martin Erwig, Danila Fedorin, and Alex Grejuc. 2018. Explaining Spreadsheets with Spreadsheets (Short Paper). In *Proceedings of the 17th ACM SIGPLAN International Conference on Generative Programming: Concepts and Experiences (GPCE '18), November 5–6, 2018, Boston, MA, USA.* ACM, New York, NY, USA, 7 pages. https://doi.org/10.1145/3278122.3278136

*This work is partially supported by the National Science Foundation under the grant CCF-1717300. Work financed by European Regional Development Fund through the Operational Programme for Competitiveness and Internationalization COMPETE 2020 Programme and by National Funds through the Portuguese funding agency FCT project POCI-01-0145-FEDER-016718 and NOVA LINCS UID/CEC/04516/2013, and by FLAD project 233/2014.

GPCE '18, November 5–6, 2018, Boston, MA, USA

ACM ISBN 978-1-4503-6045-6/18/11…$15.00
https://doi.org/10.1145/3278122.3278136

1 Introduction

Studies estimate that software maintenance costs make up 60% [32] to 80% [33] of the total cost of software over its life cycle. Changing a program or even just using it requires a user to understand it [30], and that takes time and effort. Research has shown that software developers spend most of their time understanding source code [34].

Program understanding gets even more complicated when a piece of software changes its owner frequently. Studies of spreadsheet usage found that 85% of the study participants did not create the spreadsheets they had to work on themselves but received them from their colleagues [25]. The same paper reports that 70% of those users have difficulties understanding them and spend hours browsing them.

To address some of these challenges, Kankuzi extensively studied the mental models of spreadsheets users [27]. He advocates techniques and tools that reflect these mental models such as abstracting implementation details by replacing cell references in formulas through user-defined names. Other approaches in this regard have tried to support the maintenance of spreadsheets by systematizing their evolution [16–18]. However, these techniques face an uphill battle, since they depend on the widespread adoption of new languages and tools to be effective. Moreover, it is difficult to support legacy spreadsheets in this way.

Therefore, we propose to augment spreadsheets with *explanations*, which can help users better understand and thus use and maintain spreadsheets. This approach allows users to continue to work with their current spreadsheet system, and it can be retroactively applied to existing spreadsheets. In the following, we illustrate our approach with an example.

When people try to understand a spreadsheet, they commonly face the time-consuming and error-prone task of resolving cell references to make sense of what formulas in a spreadsheet do. This task is often exacerbated in spreadsheets which contain a distracting and overwhelming volume of data and by the fact that referenced cells often contain

	A	B	C	D	E	F	G
1				Payroll Spreadsheet			
2	Name	Pay Rate	Regular Hours	Overtime Hours	Regular Pay	Overtime Pay	Total
3	Adams	8.9	40	5	=B3*C3	=B3*1.5*D3	=E3+F3
4	Baker	12.55	35	0	=B4*C4	=B4*1.5*D4	=E4+F4
5	Carlton	9.6	40	2	=B5*C5	=B5*1.5*D5	=E5+F5
6	Daniels	10.2	35	0	=B6*C6	=B6*1.5*D6	=E6+F6
7							
8	Totals		=SUM(C3:C6)	=SUM(D3:D6)	=SUM(E3:E6)	=SUM(F3:F6)	=SUM(G3:G6)

Figure 1. Payroll Spreadsheet

formulas that reference other cells in different parts of the spreadsheet.

This process of "reference chasing" in spreadsheets is necessary, since references are specified using row and column indexes, which do not provide any information about the value or meaning of the referenced cell. Therefore, to make sense of what a reference represents, the user has to scan the spreadsheet for the indicated row and column. This may require a user to scroll a long distance and in many cases jump across multiple cells to make sense of the initial reference.

We address this problem through the use of *label abstraction* in our spreadsheet explanation model where possible labeling information is used to replace raw index references with the values of cells that label them. In doing so, formulas that use references become more clear.

Consider Figure 1, which shows the formula view of a spreadsheet with payroll information for employees within a company. The spreadsheet was adapted from a study on spreadsheet error detection and correction [8] and is a simplified version of a real world scenario.

Applying label abstraction to the payroll spreadsheet results in the spreadsheet shown in Figure 2. Label abstraction increases readability by giving context to computation, especially as formula complexity increases.

We can observe that while the values in rows 3-6 differ, the formulas with labels are all identical. Such a pattern occurs quite frequently. The redundancy that results from the repetition is rather distracting. On the other hand, the different values in the different rows do not contribute much to the understanding of the represented computation.

Therefore, we apply another transformation to obtain an explanation of the original spreadsheet that represents repeated (groups of) rows by a single (group of) row(s). The result of this compression, which we call a *zoom*, can be seen in Figure 2 where rows 3-6 from the original spreadsheet have been compressed into one row. Such compressed rows contain two kinds of information. First, columns (such as E-G) that contain in the uncompressed sheet a single repeated formula contain just that single formula. Second, columns (such as A-D) with different values in different rows contain a range that captures all values found in the respective column. The same zoom compression technique can, of course, be applied to repeated columns.

Note that both transformations preserve the essential structure and the key computing elements of the original spreadsheet. We call a spreadsheet that is the result of label abstraction and zoom compression an *explanation sheet*.

In the remainder of this paper, we first present some general design principles for explanations in Section 2. Based on these principles, we have developed our spreadsheet explanation model, which is formalized in Section 3. In Section 4 we present a preliminary artifact evaluation, followed by a small user study in Section 5. We discuss related work in Section 6 and present some conclusions in Section 7.

2 Explanation Principles

Informed in part by previous work on explanations [20, 35], but also by the experience during the creation of spreadsheet explanations (see Section 4) we have identified a number of general principles that we believe should guide the development of explanations. We present these principles here in a separate section, so that they can also inform the design of explanations for other languages.

In general, explanations can take on many different forms. Taking a programming language perspective, we have found it useful to conceptualize an explanation system as consisting of two languages: (A) the language whose programs are to be explained and (B) the language in which explanations are expressed. We call the former the *subject language* and the latter the *explanation language*. Correspondingly, we call programs of the subject language *subject programs* (or *programs* for short), and we call programs of the explanation language *explanation programs* (or *explanations* for short). In the context of spreadsheet explanations, this means that we refer to spreadsheets sometimes as subject (spread)sheets and that we call their explanations explanation (spread)sheets.

The following four principles have emerged as guidelines for the design of spreadsheet (and other) explanations.

(1) **Structure Preservation**. *An explanation language should retain key subject language structures.* Subject language structures can provide easy access to an explanation, since users are already familiar with these structures. Moreover, reused structures facilitate the alignment of explanations with subject programs.
(2) **Abstraction**. *An explanation language should aim at high-level descriptions that abstract from details of the subject language.* Abstraction makes explanations faster to absorb. It also allows explanations to provide summaries of subject programs.
(3) **Partiality**. *An explanation language should support partial explanations.* In other words, an explanation should not be required to cover all of a subject program. Partiality supports a gentle slope approach to explanations, since it allows the incremental construction of more and more complete explanations. Moreover, partiality allows one to ignore parts that cannot be explained (because they are not understood) or are trivial or unimportant.

	A	B	C	D	E	F	G
1				Payroll Spreadsheet			
2	Name	Pay Rate	Regular Hours	Overtime Hours	Regular Pay	Overtime Pay	Total
3	Adams	8.9	40	5	=Pay Rate*Regular Hours	=Pay Rate*1.5*Overtime Hours	=Regular Pay+Overtime Pay
4	Baker	12.55	35	0	=Pay Rate*Regular Hours	=Pay Rate*1.5*Overtime Hours	=Regular Pay+Overtime Pay
5	Carlton	9.6	40	2	=Pay Rate*Regular Hours	=Pay Rate*1.5*Overtime Hours	=Regular Pay+Overtime Pay
6	Daniels	10.2	35	0	=Pay Rate*Regular Hours	=Pay Rate*1.5*Overtime Hours	=Regular Pay+Overtime Pay
7							
8	Totals		=SUM(Regular Hours)	=SUM(Overtime Hours)	=SUM(Regular Pay)	=SUM(Overtime Pay)	=SUM(Total)

	A	B	C	D	E	F	G
1				Payroll Spreadsheet			
2	Name	Pay Rate	Regular Hours	Overtime Hours	Regular Pay	Overtime Pay	Total
3	*[Adams...Daniels]*	*[8.9...12.55]*	*[35...40]*	*[0...5]*	=Pay Rate*Regular Hours	=Pay Rate*1.5*Overtime Hours	=Regular Pay+Overtime Pay
4							
5	Totals		=SUM(Regular Hours)	=SUM(Overtime Hours)	=SUM(Regular Pay)	=SUM(Overtime Pay)	=SUM(Total)

Figure 2. *Top*: Payroll Spreadsheet with Label Abstraction; *Bottom*: Explanation Sheet for the Payroll Spreadsheet

(4) **Compositionality**. *An explanation language should support constructing bigger explanations from smaller ones.* This requires composition operators for explanations. Compositionality supports the systematic construction of explanations and the reuse of explanations. Together with partiality, compositionality supports the distributed creation of explanations by different people who understand different parts of the subject program.

Note that the first two principles are sometimes in conflict with each other because abstraction calls for ignoring structures in the subject language. Moreover, there is a trade-off between the benefits that can be gained from abstraction and the explicitness and simplicity offered by a detailed and concrete description of a computation. As illustrated in [20], this problem can be addressed by providing for one subject program a set of explanations that are related and can be explored in a systematic way.

3 An Explanation Language for Spreadsheets

In Sections 3.1 and 3.2 we define spreadsheets and explanation sheets, respectively. In Section 3.3 we introduce a relationship between the two languages that captures the notion of explainability.

3.1 Spreadsheets

A spreadsheet is a rectangular grid of cells that contain formulas and values. We can represent spreadsheets $s \in S$ as partial mappings from addresses $A = \mathbb{N} \times \mathbb{N}$ to formulas. Formulas are either plain values ($v \in Val$), application of operations (ω) to other formulas, and references to cells ($a \in A$). The set of values includes an empty value $\sqcup$, which allows us to distinguish undefined cells that are part of the spreadsheet from undefined cells on the outside.

$$f \in Fml \quad ::= \quad v \mid \omega(f, \ldots, f) \mid a$$

Abstracting from the contents of cells, we use the type constructor $\boxplus_\alpha = A \to \alpha$ to represent sheets indexed by addresses and storing values of type α. A spreadsheet $\boxplus_{Fml}$ is then simply a sheet of formulas. Formulas are evaluated to values *Val*, and we call the result of the evaluation of a spreadsheet a *value sheet*, which is a sheet of values $\boxplus_{Val}$. The semantics of a spreadsheet language are given by a function $[\![\cdot]\!] : \boxplus_{Fml} \to \boxplus_{Val}$ that maps spreadsheets to value sheets.

3.2 Explanation Sheets

Following the structure preservation principle from Section 2, we design an explanation of a spreadsheet to be itself a kind of spreadsheet, a so-called *explanation sheet* that stores formula explanations in cells. Following the abstraction principle, an explanation sheet should abstract from some of the details of the spreadsheet and should thus be smaller in size. We therefore need a definition that allows one cell in an explanation sheet to explain many cells in a spreadsheet.

To explain formulas, we need explanations for values, references, and expressions built by operations applied to other formulas. Since the values in a spreadsheet are either numbers or strings, which are both ordered domains, we can summarize a set of values from different cells by a *value range* ($\bar{v} \in \overline{Val} = Val \times Val$). Similarly, a set of references can be summarized by an *address range* ($\bar{a} \in \overline{A} = A \times A$). The two addresses of a range represent opposing corners of a rectangular area, and the region denoted by a range is given by the function $\rho : \overline{A} \to \mathcal{A}$, which is defined as follows ($\downarrow/\uparrow$ compute the minimum/maximum of two numbers).

$$\rho(((x_1, y_1), (x_2, y_2)) = \{(x, y) \mid x_1{\downarrow}x_2 \le x \le x_1{\uparrow}x_2 \\ \wedge\, y_1{\downarrow}y_2 \le y \le y_1{\uparrow}y_2\}$$

Since labels have been successfully employed in the past for annotating and explaining cells [4, 19, 26, 29], we use labels to explain a set of references by one or two (row and/or column) labels $\ell \in Lab = Val \cup Val \times Val$. More precisely, based on a relationship $\mathcal{L} \subseteq A \times A$ where $(a, a') \in \mathcal{L}$ whenever the

$$\textsc{Value}\quad v \lhd v \qquad \textsc{Value Range}\quad \frac{v_1 \le v \le v_2}{(v_1, v_2) \lhd v} \qquad \textsc{Address Range}\quad \frac{a_1 \le a \le a_2}{(a_1, a_2) \lhd a}$$

$$\textsc{Formula}\quad \frac{x_1 \lhd f_1 \quad \ldots \quad x_n \lhd f_n}{\omega(x_1, \ldots, x_n) \lhd \omega(f_1, \ldots, f_n)} \qquad \textsc{Label}\quad \frac{L(a) = \ell}{\ell \lhd a}$$

$$\textsc{Empty Value}\quad (v_1, v_2) \lhd \sqcup \qquad \textsc{Empty Formula}\quad \omega(x_1, \ldots, x_n) \lhd \sqcup \qquad \textsc{Unexplained}\quad \bot \lhd f$$

Figure 3. Formula Explanations

value $S(a)$ in cell a is considered to be a label for cell a', we can define a partial *labeling function* $L : A \to Lab$, which identifies values as labels for cells.

$$L(a') = \begin{cases} S(a) & \text{if } \mathcal{L}^{-1}(a') = \{a\} \\ (S(a_1), S(a_2)) & \text{if } \mathcal{L}^{-1}(a') = \{a_1, a_2\} \end{cases}$$

$L(a')$ is undefined whenever $\mathcal{L}^{-1}(a') = \emptyset$.

We explain sets of formulas that share a common structure and differ only in their references by a formula with labels abstracting the references. Finally, we represent unexplained areas using the special value $\bot$ ("unexplained"), which allows us to reduce potentially large chunks of a spreadsheet by a single row, column, or cell.

Thus we obtain the following definition of explanation formulas and the derived notion of explanation sheets $\boxplus_{Xpl}$.

$$x \in Xpl \quad ::= \quad v \mid \bar{v} \mid a \mid \bar{a} \mid \ell \mid \omega(x, \ldots, x) \mid \bot$$

The structure preservation embraced by $\boxplus_{Xpl}$ aligns the structure and composition of an explanation sheet with that of the explained spreadsheet.

3.3 Explaining Spreadsheets with Explanation Sheets

A spreadsheet explanation is captured by a so-called *zoom* $X \overset{\eta}{\lhd} S$, which consists of an explanation sheet X, a spreadsheet S, a total function η that embeds the spreadsheet into the explanation, that is, $dom(\eta) = dom(S) \wedge rng(\eta) = dom(X)$, and whose explanation formulas explain the formulas of the spreadsheet. The totality of η ensures that every cell in S is covered by a cell in X. We don't require zooms to be surjective to allow for "filler cells" in the explanation sheets that serve no other purpose than to turn explanation sheets into rectangular areas.

The purpose of zooms is to explain a number of similar cells by one cell. Specifically, when $\eta^{-1}(a) = \{a_1, \ldots, a_k\}$, we use cell a to summarize, or explain, all the cells $a_1, \ldots, a_k$. We can formalize this idea through the notion of *formula explanation*, which is defined as a binary relationship $x \lhd f$ that says an explanation formula x explains a spreadsheet formula f, see Figure 3.

$$[\![v]\!]_X = (v, v) \qquad [\![\bar{v}]\!]_X = \bar{v} \qquad [\![a]\!]_X = [\![X(a)]\!]_X$$

$$[\![\bar{a}]\!]_X = \updownarrow\{[\![X(a)]\!]_X \mid a \in \rho(\bar{a})\} \qquad [\![\ell]\!]_X = \updownarrow L^{-1}(\ell)$$

$$\frac{[\![x_i]\!]_X = (v_i^1, v_i^2) \qquad v_i^1 \le v_i \le v_i^2}{[\![\omega(x_1, \ldots, x_n)]\!]_X = \updownarrow\{[\![\omega(v_1, \ldots, v_n)]\!]_X\}} \qquad [\![\bot]\!]_X = \bot$$

Figure 4. Explanation Semantics

The cases for plain value, value range, and address range should be obvious. Rule Formula requires that the explanation and explained formulas have the same structure, and the premise in the rule Label ensures that a label exists. The rules Empty Value and Empty Formula allow empty values to be explained by ranges and formulas, respectively, and the rule Unexplained allows any formula to be left unexplained.

For a zoom $X \overset{\eta}{\lhd} S$ we require that every formula in X explain all formulas in S that are mapped to it, that is:

$$\forall a' \in dom(X), \forall (a, a') \in \eta : X(a') \lhd S(a)$$

Based on the semantics of spreadsheets, we can define the semantics for explanation sheets as follows. Since explanation formulas include ranges of values and addresses, they will generally evaluate to ranges of values.[1] To resolve references the semantics needs access to the explanation sheet. Since we also have to account for $\bot$ formulas, the semantics of explanation formulas is of type $[\![\cdot]\!] : Xpl \to \boxplus_{Xpl} \to \overline{Val} \cup \{\bot\}$. The definition is shown in Figure 4. We use the function $\updownarrow V = (\downarrow V, \uparrow V)$ to compute the minimally enclosing range for a set of values V. (We also use it for addresses.)

The semantics of explanation sheets is then given by the following function $[\![\cdot]\!] : \boxplus_{Xpl} \to \boxplus_{\overline{Val} \cup \{\bot\}}$.

$$[\![X]\!] = \{(a, \bar{v}_\bot) \mid (a, x) \in X \wedge [\![x]\!]_X = \bar{v}_\bot\}$$

Note that the semantics also depends on the underlying subject sheet S and a labeling relationship $\mathcal{L}$ to resolve labels (ℓ) in explanation formulas.

Next we introduce the notion of *zoom soundness*. This is essentially the $\lhd$ relationship for value ranges and values applied to whole sheets that are connected via a function η. We say that an explanation X is *sound* for a spreadsheet S under η if $[\![X]\!] \overset{\eta}{\lhd} [\![S]\!]$. This relationship captures the notion that an explanation sheet X covers all cases of the explained spreadsheet S and that the evaluation of S holds no surprises.

Now we can present our main result, which says that zooms are sound.

Theorem 3.1 (Soundness). $X \overset{\eta}{\lhd} S \implies [\![X]\!] \overset{\eta}{\lhd} [\![S]\!]$

Note that for any spreadsheet S we always can find a trivial explanation through the zoom $S \overset{id}{\lhd} S$,[2] which means that any

[1] A single value v can always be represented by a trivial range (v, v).
[2] Here *id* denotes the identity function.

spreadsheet trivially explains itself. However, such a zoom is not really useful, since it does not achieve any abstraction. Employing a straightforward ordering on zooms based on the size of the explanation sheet, we can define that a zoom $X_1 \overset{\eta_1}{\circlearrowleft} S$ achieves a *higher explanatory reduction* than a zoom $X_2 \overset{\eta_2}{\circlearrowleft} S$ if $|dom(X_1)| < |dom(X_2)|$. Note that this relationship defines a partial order, and there isn't necessarily a single smallest explanation.

We can identify a number of interesting relationships for explanations, including notions such as explanatory coverage and based on that also an explanation refinement relationship. We leave that for future work.

4 Artifact Evaluation

We employed real-world spreadsheets from two spreadsheet corpora in the design and evaluation of our approach.

4.1 Guiding the Design of Explanations

The design of our spreadsheet explanations was guided in part by real-world example spreadsheets. With a preliminary definition of explanation formulas and zooms we set out to explain existing spreadsheets from two repositories.

Specifically, we analyzed 20 randomly selected spreadsheets from [31], which are generally well-designed spreadsheets created by experts, plus 20 randomly selected spreadsheets from the Enron spreadsheets corpus [22], which includes more than 15,000 spreadsheets.

We manually created an explanation spreadsheet for each of the 40 spreadsheets. The main purpose of this exercise was to see whether explanation formulas are general enough or maybe even unnecessarily too general and whether our definition of zooms worked as anticipated.

During this testing phase, the explanation model was revised several times. Specifically, we removed a number of explanation formulas that we originally thought to be useful because the anticipated situations did either not occur at all or only once or twice and thus were not justifying a more elaborate notion of explanation formulas. We also simplified the definition of zooms, which originally were defined recursively allowing for nested zooms to explain nested loop structures in spreadsheets. But since such a nested loop structure occurred only in one of the selected examples, we traded the more general definition for a simpler one.

4.2 Applicability and Impact

We randomly selected a new set of spreadsheets from the two sources and then used 41 worksheets from 36 different spreadsheets to analyze the applicability and effect of spreadsheet explanations.

We observed that 78% (32/41) of worksheets contained areas that could be compressed and explained by zooms (20/10/2 worksheets contained row/column/row and column zooms). Ignoring three huge spreadsheets that were basically used as databases and that would lead to a misleadingly high average, the average (min/max) size compression achieved by zooms was 64% (25%/99%).

To verify that the generated explanations were correct, we developed an explanation checker that implements the definitions from Section 3.3, specifically Figure 3. This explanation checker was applied to all explanations and helped to correct at least one error in 24 of the 41 explanation sheets. Most of the errors were due to simple typos, but the checker also found several incorrect range mappings in zooms and other reference errors. A summary of the kinds of errors that were detected is shown in Figure 5.

5 User Evaluation

In this section we describe a preliminary user study we performed to assess the usefulness of explanation spreadsheets.

Spreadsheets. For this study, we have semi-randomly selected 4 spreadsheets (labeled A through D) from 3 different sources. The selection process was semi-random in the sense that we had to randomly re-select a new spreadsheet if the previous one was not suited (for example, when it lacked formulas). We selected two spreadsheets from the EUSES corpus [21], one from [31], and one from the Enron corpus [22].

Participants. We recruited ten participants from two universities who were either computer science graduate students or had already finished their graduate degree. Most of the participants are quite experienced spreadsheet users, some of them having created over 100 spreadsheets. Two of the participants were females and eight were males with ages ranging from 23 to 45.

Procedure. For each spreadsheet we created the corresponding explanation. We then created two sets of artifacts, each consisting of two spreadsheets and two explanation sheets. The first set contained the spreadsheets for A and C and the explanation sheets for B and D, and the second set contained the spreadsheets for B and D and the explanation sheets for A and C. We then randomly assigned each participant one of the sets. This way each participant had to work with spreadsheets as well as explanation sheets, in different orders, thus reducing learning effects. We had the participants go over a one-page tutorial about spreadsheet explanations before answering the actual questions. For each of the four spreadsheets we asked two questions:

N	Error Type
4	Value (value in S is mapped to a different value in X)
7	Range (value in S is not covered by the range in X)
4	Reference (undefined references in mappings)
1	Label A (value of the label does not match the value in S)
8	Label B (labeling does not correctly abstract reference)

Figure 5. Errors found by the Explanation Checker

Table 1. Average times and scores in the empirical study

		average time		average score	
		subject	explanation	subject	explanation
A	Q1	1.3	2.1	2.2	2.4
	Q2	1.1	2.2	3.0	2.8
B	Q1	3.1	2.9	2.0	2.6
	Q2	2.5	3.7	2.0	1.8
C	Q1	2.1	1.8	3.0	1.0
	Q2	1.0	2.9	2.4	1.4
D	Q1	3.6	5.4	1.2	1.4
	Q2	6.8	3.3	1.8	2.0

Q1 **What** is being calculated in row/column/cell X?
Q2 **How** are the values in row/column/cell X calculated?

Results. We present the results of the study in Table 1. For each spreadsheet and its explanation we show the average time (in minutes) participants took to answer each question and the average score of the answers. We scored each answer with a value from 0 (wrong answer) to 3 (entirely correct).

Discussion. The user evaluation produced mixed results. Explanation sheets led to higher scores in 3 out of the 4 scenarios, with the exception of C, which produced significantly lower results. Here we note that the explanation sheet for C employed a column header (S) as a label where none was provided by the subject spreadsheet. As the participants had no prior knowledge of explanations, this could have made it hard to infer the meaning of the column reference, thus impacting understandability.

There is no significant difference between the average times it took participants to answer the questions. With the exception of D, participants were able to determine *how* a computation was performed faster using the explanation. However, explanations were only faster at explaining *what* a computation calculated in cases B and C.

Interestingly, participants took longer to answer questions for simple spreadsheets using explanations. This can possibly be attributed to the fact that the participants have had extensive experience with spreadsheets, while none with explanations. This also seems to indicate that explanation sheets are probably more useful for complex spreadsheets.

These results indicate the potential of explanation sheets for providing a better understanding of spreadsheets. In their answers to a post-study survey eight out of ten participants said that they found explanation sheets somewhat or very helpful. (For the two other participants they didn't make a difference.) Also, eight of the participants would want to use explanation sheets in the future.

6 Related Work

Amalfitano et al. have developed a tool to help end users to comprehend VBA-based Excel applications [6]. This tool is designed to work with Excel programs that heavily depend on VBA macros, explaining the relation between the macros and the cells, and cannot be used to understand the data or cell computations in a spreadsheet as we propose.

Kankuzi and Sajaniemi have investigated spreadsheet authors' mental models [27]. Based on that, they proposed a tool that translates cell references into domain/real-world terms using the labels in the spreadsheet. However, their focus is on error detection, while ours in on understanding the content of a spreadsheet.

Several authors have proposed techniques to identify structural information in spreadsheets [2, 5, 7, 11–15, 23]. Although identified structures may help understand spreadsheets, there is no direct evidence for this, and none of the mentioned approaches was meant to help understanding spreadsheets, but instead to give some kind of support to manipulate spreadsheet content.

The use of spreadsheets labels is not new [19]. In fact, several approaches have been proposed for the inference of cell labels [1], to reason about spreadsheets [10] using their labels, and to use them for error checking [3]. And while sometimes labels are used to explain spreadsheet errors [4], none of this work was intended to explain spreadsheets purpose. Some tools provide *named ranges* for defining a name for a range of cells, which can then be used instead of regular references. Although this feature has been shown to make debugging less effective [28], there is no evidence of its impact in understanding spreadsheets.

Hermans has proposed techniques to understand the dependencies between different worksheets [24, 25]. In contrast, our work improves the understandability of each worksheet. Thus, these two approaches are complementary.

Some researchers have proposed tools to support users in documenting their spreadsheets [9]. However, such tools require users to write documentation, which they usually do not do. While our approach also relies on an additional artifact, this has a familiar structure and can be added incrementally. Most importantly, the formal structure of explanation sheets and their relationship to spreadsheets offers opportunities for the automatic inference of explanation sheets, which is something we plan to work on in the future.

7 Conclusions

We have presented the concept of explanation sheets to support the understanding of spreadsheets. The design of explanation sheets has some appealing properties: First, they facilitate the gradual and incremental construction of spreadsheet explanations. Second, their formalization supports the definition of tools for checking, for example, the correctness of explanation, which we have already exploited. There are many other support tools that can be envisioned and that we will investigate in future work, such as checking the coverage of explanations, checking the compatibility of alternative explanations, and the inference of explanation sheets.

References

[1] R. Abraham and M. Erwig. 2004. Header and Unit Inference for Spreadsheets Through Spatial Analyses. In *IEEE Int. Symp. on Visual Languages and Human-Centric Computing*. 165–172.

[2] R. Abraham and M. Erwig. 2006. Inferring Templates from Spreadsheets. In *28th IEEE Int. Conf. on Software Engineering*. 182–191.

[3] R. Abraham and M. Erwig. 2007. UCheck: A Spreadsheet Unit Checker for End Users. *Journal of Visual Languages and Computing* 18, 1 (2007), 71–95.

[4] R. Abraham, M. Erwig, and S. Andrew. 2007. A Type System Based on End-User Vocabulary. In *IEEE Int. Symp. on Visual Languages and Human-Centric Computing*. 215–222.

[5] Sorin Adam and Ulrik Pagh Schultz. 2015. Towards Tool Support for Spreadsheet-based Domain-specific Languages. *SIGPLAN Not.* 51, 3 (Oct. 2015), 95–98.

[6] Domenico Amalfitano, Vincenzo De Simone, Anna Rita Fasolino, and Porfirio Tramontana. 2016. EXACT: A tool for comprehending VBA-based Excel spreadsheet applications. *Journal of Software: Evolution and Process* 28, 6 (2016), 483–505.

[7] Domenico Amalfitano, Anna Rita Fasolino, Porfirio Tramontana, Vincenzo De Simone, Giancarlo Di Mare, and Stefano Scala. 2015. A Reverse Engineering Process for Inferring Data Models from Spreadsheet-based Information Systems: An Automotive Industrial Experience. In *Data Management Technologies and Applications*. 136–153.

[8] Brian Bishop and Kevin McDaid. 2008. An Empirical Study of End-User Behaviour in Spreadsheet Error Detection & Correction. *CoRR* abs/0802.3479 (2008).

[9] Diogo Canteiro and Jácome Cunha. 2015. SpreadsheetDoc: An Excel Add-in for Documenting Spreadsheets. In *Proceedings of the 6th National Symposium of Informatics*.

[10] C. Chambers and M. Erwig. 2010. Reasoning about Spreadsheets with Labels and Dimensions. *Journal of Visual Languages and Computing* 21, 5 (2010), 249–262.

[11] Zhe Chen and Michael Cafarella. 2013. Automatic Web Spreadsheet Data Extraction. In *3rd International Workshop on Semantic Search Over the Web*. Article 1, 8 pages.

[12] J. Cuna, M. Erwig, and J. Saraiva. 2010. Automatically Inferring ClassSheet Models from Spreadsheets. In *IEEE Int. Symp. on Visual Languages and Human-Centric Computing*. 93–100.

[13] Jácome Cunha, Martin Erwig, Jorge Mendes, and João Saraiva. 2016. Model inference for spreadsheets. *Automated Software Engineering* 23, 3 (2016), 361–392.

[14] Jácome Cunha, João Saraiva, and Joost Visser. 2009. From Spreadsheets to Relational Databases and Back. In *2009 ACM SIGPLAN Symposium on Partial Evaluation and Semantics-based Program Manipulation*. 179–188.

[15] Wensheng Dou, Shi Han, Liang Xu, Dongmei Zhang, and Jun Wei. 2018. Expandable Group Identification in Spreadsheets. In *33rd ACM/IEEE Int. Conf. on Automated Software Engineering*. 498–508.

[16] G. Engels and M. Erwig. 2005. ClassSheets: Automatic Generation of Spreadsheet Applications from Object-Oriented Specifications. In *20th IEEE/ACM Int. Conf. on Automated Software Engineering*. 124–133.

[17] M. Erwig, R. Abraham, I. Cooperstein, and S. Kollmansberger. 2005. Automatic Generation and Maintenance of Correct Spreadsheets. In *27th IEEE Int. Conf. on Software Engineering*. 136–145.

[18] M. Erwig, R. Abraham, S. Kollmansberger, and I. Cooperstein. 2006. Gencel – A Program Generator for Correct Spreadsheets. *Journal of Functional Programming* 16, 3 (2006), 293–325.

[19] M. Erwig and M. M. Burnett. 2002. Adding Apples and Oranges. In *4th Int. Symp. on Practical Aspects of Declarative Languages (LNCS 2257)*. 173–191.

[20] M. Erwig and E. Walkingshaw. 2013. A Visual Language for Explaining Probabilistic Reasoning. *Journal of Visual Languages and Computing* 24, 2 (2013), 88–109.

[21] Marc Fisher and Gregg Rothermel. 2005. The EUSES Spreadsheet Corpus: A Shared Resource for Supporting Experimentation with Spreadsheet Dependability Mechanisms. *SIGSOFT Softw. Eng. Notes* 30, 4 (May 2005), 1–5.

[22] Felienne Hermans and Emerson Murphy-Hill. 2015. Enron's Spreadsheets and Related Emails: A Dataset and Analysis. In *37th Int. Conf. on Software Engineering*. 7–16.

[23] Felienne Hermans, Martin Pinzger, and Arie van Deursen. 2010. Automatically Extracting Class Diagrams from Spreadsheets. In *European Conference on Object-Oriented Programming 2010*. 52–75.

[24] Felienne Hermans, Martin Pinzger, and Arie van Deursen. 2011. Breviz: Visualizing Spreadsheets using Dataflow Diagrams. *CoRR* abs/1111.6895 (2011).

[25] Felienne Hermans, Martin Pinzger, and Arie van Deursen. 2011. Supporting Professional Spreadsheet Users by Generating Leveled Dataflow Diagrams. In *33rd Int. Conf. on Software Engineering*. 451–460.

[26] B. Kankuzi and J. Sajaniemi. 2014. Visualizing the problem domain for spreadsheet users: A mental model perspective. In *IEEE Symp. on Visual Languages and Human-Centric Computing*. 157–160.

[27] Bennett Kankuzi and Jorma Sajaniemi. 2016. A mental model perspective for tool development and paradigm shift in spreadsheets. *International Journal of Human-Computer Studies* 86 (2016), 149 – 163.

[28] Ruth McKeever, Kevin McDaid, and Brian Bishop. 2009. An Exploratory Analysis of the Impact of Named Ranges on the Debugging Performance of Novice Users. *CoRR* abs/0908.0935 (2009).

[29] B. A. Nardi and J. R. Miller. 1991. Int. Journal of Man-Machine Studies. (1991), 161–184.

[30] Santanu Paul, Atul Prakash, Erich Buss, and John Henshaw. 1991. Theories and Techniques of Program Understanding. In *Conf. of the Centre for Advanced Studies on Collaborative Research*. 37–53.

[31] Stephen G. Powell and Kenneth R. Baker. 2003. *The Art of Modeling with Spreadsheets*. John Wiley & Sons, Inc., New York, NY, USA.

[32] R. Pressman. 2001. *Software Engineering: A Practitioner's Approach (5th ed.)*. McGraw-Hill, New York, NY.

[33] C. Verhoef. 2000. How to Implement the Future. In *26th Euromicro Conference*. 32–47.

[34] A. von Mayrhauser, M. Vans, and A. Howe. 1997. Understanding Behaviour During Enhancement of Large-scale Software. *Journal on Software Maintenance: Research and Practice* 9, 5 (1997), 299–327.

[35] E. Walkingshaw and M. Erwig. 2011. A DSEL for Studying and Explaining Causation. In *IFIP Working Conference on Domain-Specific Languages*. 143–167.