Polaris: Providing Context Aware Navigation in Spreadsheets

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

***Abstract*—Spreadsheets are a successful example of an end-user programming language, and the spreadsheet paradigm shares several characteristics like composition, selection, and repetition with programming languages. There are compelling reasons that spreadsheets are code. For most programming languages, developers are supported by powerful IDEs. However, spreadsheets are missing such an IDE. In our current work we are researching how spreadsheet users could be supported by an IDE for spreadsheets and what kind of functionality should be included? As a preliminary result of this research, we introduce in this showpiece: Polaris, an excel Add-in that provides users with context-aware navigation in spreadsheets.**

I. Background and Motivation

A. Spreadsheets are error-prone

Previous research shows that spreadsheets are error-prone [1]. Initially, we believed that an explanation for this error-proneness is that spreadsheet models are very complex. This complexity could be caused by a high degree of coupling between cells and worksheets and the use of complicated formulas. However, based on the analysis of several large spreadsheet corpora we learned that the majority of spreadsheets are not complex [2], [3].

After ruling out complexity as a possible explanation we directed our attention to the user interface. Users enter their formulas in single cells. In these formulas, they can reference other cells either on the same worksheet or on a different worksheet. When the user has entered the formula, the spreadsheet interface hides the formula and instead displays the result of the formula. Because of this, not only the formulas are hidden, but also the implicit relations between the different cells. After entering several formulas it becomes more and more difficult to understand how the different parts of the spreadsheet model are related. The design of the spreadsheet is hidden behind the spreadsheet itself. The spreadsheet interface makes it more difficult for the user to comprehend the spreadsheet.

B. Spreadsheets are code

There are several compelling reasons that spreadsheets are code, and that developing spreadsheet models is similar to programming [4]. However, there are also differences. Most programming languages are supported by powerful Integrated Development Environments (IDEs). They support users for example with navigation tools, smell and clone detections, testing, refactoring, etc. Unfortunately, the spreadsheet interface is not as powerful as these IDEs. In our current research, we are analyzing if we can make the development of spreadsheet models easier by providing spreadsheet users with an IDE where they can build the model with a visual language [5]. As a preliminary result of this research, we present in this showpiece Polaris[1], an Excel Add-In for context-aware navigation in spreadsheets.

C. Information Organization in Spreadsheets

One of the focus points in our current research, related to the IDE and visual language, is the organization of information in spreadsheets. From the results of a controlled experiment with spreadsheet users, we learned that they prefer to divide their data over multiple worksheets within a single spreadsheet [6]. We also encountered this habit when analyzing spreadsheet corpora. Furthermore, it is one of the recommendations of the FAST standard: a collection of guidelines for building spreadsheet models. The standard advises to: *'separate worksheets by functional class'* [7].

From an information organization point of view, this is sound advice, but it inevitably leads to more formulas with connections to other worksheets. If a user wants to change or debug such a formula, he has to switch from one sheet to another. By default, when switching worksheets, Excel displays the last selected cell on that sheet. This means that the user has to orient himself and navigate to the cell he wanted to inspect when he decided to switch worksheets. We hypothesize that the act of reorienting and navigating to the right cell interrupts the line of thought of the user and increases the chances of introducing an error and makes it more difficult to understand the formula. This problem is not unique for spreadsheet developers. When writing code, developers have to switch between different files and encounter similar problems. Improved navigation support, like Code Bubbles [8] and Code Canvas [9], have been proposed to solve this problem in the IDE.

D. Polaris

To make switching between worksheets easier for the user, we developed Polaris. It is an add-in that monitors which cell the user is inspecting and based on that information will

[1]Both the add-in and an accompanying video are available for download at: *add-in: https://dx.doi.org/10.6084/m9.figshare.3474908, video: https://dx.doi.org/10.6084/m9.figshare.3474875*

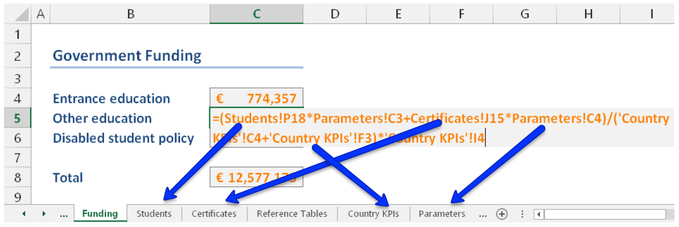
‘guess’ to which cell it should navigate when a user switches from one sheet to another.

This is a different approach than the Code Bubbles implementation. We choose this different approach because the mixing of code and data in spreadsheets makes it challenging to automatically determine which fragment of a spreadsheet could be represented in a single bubble without harming the overall comprehensibility of the spreadsheet model.

II. Implementation

Polaris has been developed using Visual Basic for Applications (VBA) as an add-in that can be added to the Excel interface. The context aware navigating is not useful in every scenario (for example during data entry) and therefore the user has the option to toggle it on and off.

We discuss the implementation of Polaris using the example in Figure 1. In this example, the user is analyzing the formula in cell C5. The cells used in this formula are located on four different worksheets: *Students*, *Parameters*, *Certificates*, and *Country KPIs*.



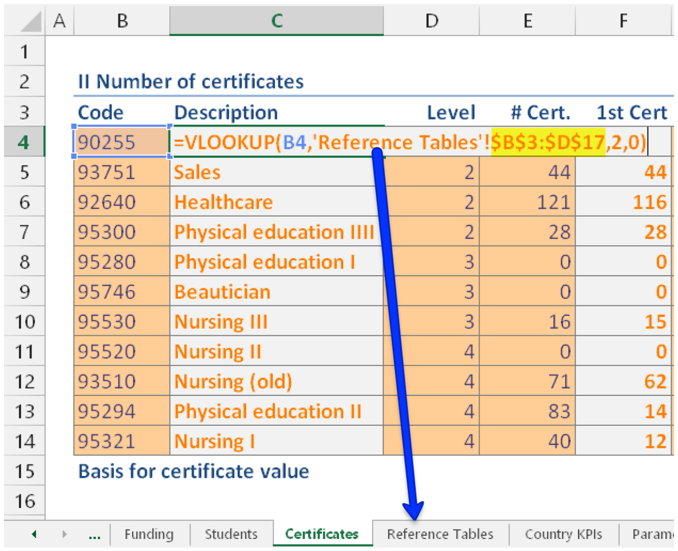
	A	B	C	D	E	F	G	H	I
1									
2		Government Funding							
3									
4		Entrance education	€ 774,357						
5		Other education	=(Students!P18*Parameters!C3+Certificates!J15*Parameters!C4)/('Country KPIs'!C4+'Country KPIs'!F3)*'Country KPIs'!I4						
6		Disabled student policy							
7									
8		Total	€ 12,577,1[illegible]						
9									

Fig. 1. Example of a formula that makes reference to multiple worksheets

Polaris will track the last cell that was selected by the user (in this case cell C5 on worksheet *Funding*). As soon as the user activates a different sheet, for example *Students*, Polaris will analyze the references in the formula of C5 to see if one or more of these references are located on sheet *Students*. This is the case for the first reference: Students!P18. It is also the only reference that is located on the sheet *Students* and therefore Polaris activates cell P18. If cell C5 would not have contained any reference that was located on sheet *Students* it would have fallen back to the default and activated the last selected cell on sheet *Students*.

A different situation occurs when the user decides to switch to sheet *Country KPIs* for which three references are made in this formula: ‘Country KPIs’!C4, ‘Country KPIs’!F3, and ‘Country KPIs’!I4. Now Polaris has to determine which of these cells will be shown to the user. If it is possible to show all references simultaneously on the same screen, Polaris will do so. If not, it will show the first reference in the formula that is located on this sheet. If the user wants to go the next operand it can simply use the key combination alt +] and Polaris will then navigate to that cell.

In the above examples, the references to the other worksheets were references to a single cell. However, formulas can also contain a reference to a range of cells. An example of this can be found in Figure 2. In this example, the formula in cell C4 on the sheet Certificates is making a reference with one of



	A	B	C	D	E	F
1						
2		II Number of certificates				
3		Code	Description	Level	# Cert.	1st Cert
4		90255	=VLOOKUP(B4,'Reference Tables'!$B$3:$D$17,2,0)			
5		93751	Sales	2	44	44
6		92640	Healthcare	2	121	116
7		95300	Physical education IIII	2	28	28
8		95280	Physical education I	3	0	0
9		95746	Beautician	3	0	0
10		95530	Nursing III	3	16	15
11		95520	Nursing II	4	0	0
12		93510	Nursing (old)	4	71	62
13		95294	Physical education II	4	83	14
14		95321	Nursing I	4	40	12
15		Basis for certificate value				
16						

Fig. 2. Example of a formula containing a reference to a range of cells on a different worksheet

it’s operands to the cells B3 up to and including D17 on sheet ’Reference Tables’. In this situation, Polaris will check if it is possible to display the full range on a single screen. If that is not possible it will activate the top-left cell of the range.

In future research, we will extend the functionality of Polaris to context-aware navigation within a single worksheet. Furthermore, we will conduct an experiment with spreadsheet users to validate the usefulness of the tool.

References

[1] R. R. Panko, “What we know about spreadsheet errors,” *Journal of Organizational and End User Computing (JOEUC)*, vol. 10, no. 2, pp. 15–21, 1998.

[2] B. Jansen, “Enron versus euses: A comparison of two spreadsheet corpora,” in *Proceedings of the 2nd Workshop on Software Engineering Methods in Spreadsheets, Florence, Italy*, 2015.

[3] B. Jansen and F. Hermans, “Code smells in spreadsheet formulas revisited on an industrial dataset,” in *Proceedings of the 2015 International Conference on Software Maintenance and Evolution*. IEEE Press, 2015, pp. 372–380.

[4] F. Hermans, B. Jansen, S. Roy, E. Aivaloglou, A. Swidan, and D. Hoepelman, “Spreadsheets are code: An overview of software engineering approaches applied to spreadsheets,” in *Proceedings of the 23rd IEEE International Conference on Software Analysis, Evolution, and Reengineering*, 2016.

[5] B. Jansen and F. Hermans, “Using a visual language to create better spreadsheets,” *Software Engineering Methods in Spreadsheets*, 2014.

[6] B. Jansen and F. Hermans, “The effect of delocalized plans on the maintainability of spreadsheets - a controlled experiment,” Delft University of Technology - Software Engineering Research Group, Tech. Rep. 014, 2016.

[7] FAST Standard Organisation. (2015) The fast standard - practical, structured design rules for financial modelling, version fast02a. [Online]. Available: http://www.fast-standard.org/

[8] A. Bragdon, S. P. Reiss, R. Zeleznik, S. Karumuri, W. Cheung, J. Kaplan, C. Coleman, F. Adeputra, and J. J. LaViola Jr, “Code bubbles: rethinking the user interface paradigm of integrated development environments,” in *Proceedings of the 32nd ACM/IEEE International Conference on Software Engineering-Volume 1*. ACM, 2010, pp. 455–464.

[9] R. DeLine and K. Rowan, “Code canvas: zooming towards better development environments,” in *Proceedings of the 32nd ACM/IEEE International Conference on Software Engineering-Volume 2*. ACM, 2010, pp. 207–210.