Proposing a Visual Language for Spreadsheets

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

I. Introduction

Spreadsheets are extensively used by companies. Information embedded in these spreadsheets often forms the basis for business decisions. However, spreadsheets are known to be error-prone and therefore, the risk exists that important business decisions could be made on inaccurate information.

The success of spreadsheets can be partially explained by their easy-to-use interface. It is this same interface, nevertheless, that is responsible for part of the problems that are associated with spreadsheets. In spreadsheet software, it is easy to enter data and formulas. When entered, only the result of the formula but not the formula itself is presented to the user. The underlying design of the spreadsheet is hidden 'behind' itself and this makes it difficult to understand it.

Another factor that affects the understandability of a spreadsheet is the way the information is structured. In industry best practices for the optimal structure of a spreadsheet have been developed.[1] A common design guideline is to split input, model and output. This design principle works well if input and calculations are relatively stable. For example, a spreadsheet that calculates the value of a company based on the input coming from their financial statements. The principle is, however, less suitable if a user intends to analyze the effect of different input values. For example, if a user wants to analyze the effects of different interest rates or cash flows on the return of investment in a business calculation for a new product line in a factory, it is more convenient to have input and output values close together. Thus, while guidelines exist, the optimal way to structure a spreadsheet depends on the kind of problem it has to solve.

In practice, users do not make a conscious choice how to structure their spreadsheets. Even if they do have the knowledge, it is often difficult to determine the right structure in advance. Invited by the interface, users enter their data and formulas without giving the structure a lot of thought. When the complexity of spreadsheets increases over time they end up with a disorganized model. At that point it is difficult to change the underlying structure and the risk of errors is imminent.

This brings us to the three problems we focus on:

1) The design of a spreadsheet is hidden behind the spreadsheet, increasing the difficulty for a user to get an overview of the design
2) Users are not aware of the best way to structure a spreadsheet
3) It is difficult to determine the right structure upfront. Therefore, users start modeling without making conscious decisions about the structure of the spreadsheet. The model works, but is often poorly structured and error-prone.

In previous work [1] we introduced these problems in more detail. In this extended abstract we share some initial results and present a research plan for future work.

[1] See for example http://www.fast-standard.org

II. Background

There have been previous attempts to address this. In 2001 Burnett *et al.* [2] developed Forms/3, a general purpose visual programming language that is consistent with the spreadsheet paradigm. Two principles in particular guided the development process: directness and immediate visual feedback. Instead of developing a general purpose programming language, we aim to develop a visual language that is designed to generate well formed spreadsheets, while honoring the concepts of directness and immediate visual feedback.

Engels and Erwig [3] have described an automatic transformation process to generate a spreadsheet from a ClassSheet (an object-oriented model for spreadsheet applications). Cunha *et al.* [4] have further improved the concept of ClassSheets. They have embedded the ClassSheet spreadsheet models in spreadsheets themselves. The authors have also presented a technique to perform co-evolution of the ClassSheet model and the related spreadsheet. Modifications to the model are automatically propagated to the spreadsheet.

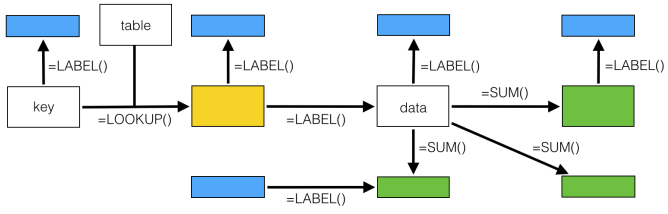
The main difference between the ClassSheet approach and ours is that our visual language does not use the tabular two-dimensional layout of spreadsheet design. If the model is represented in a spreadsheet-like layout, the design remains hidden. Furthermore, we believe that spreadsheet users - who are not professional programmers - should not be required to have knowledge of object-oriented principles.

By using ClassSheets the users have to adhere to strict layout rules when building their spreadsheets. Yet we still believe that users should be able to freely influence the spreadsheet's layout. Spreadsheets are often used for financial reporting [5] and layout is an important factor for effective reporting.

III. Introducing a Visual Language

To address the problems described in Section I, we envision an alternative user interface for the development of a spreadsheet. The basis for this user interface is a visual language (see Figure 1). By introducing an alternative interface we are facing

the challenge of user adoption. One of the success factors of spreadsheets is their flexibility and ease of use. If users have to learn a specific programming language before they can start developing a spreadsheet, we expect that the adoption of this alternative user interface will be very low. If we can, however, develop a visual language that is easy to understand, works intuitively and at the same time makes use of a drag and drop interface, we expect a higher adoption. It is not uncommon for business users to work with programs like Microsoft Visio to document business processes or using a graphical interface to build reports in a business intelligence system. We aim to give the visual language an interface that is similar.



Preliminary sketch of visual language

	A	B	C	D	E	F	G	H
1								
2		AccountNr	Description	Q1	Q2	Q3	Q3	Total
3		80000	Sales Revenu	€ 9.091	€ 7.124	€ 6.873	€ 6.694	€ 29.782
4		30000	Cost of Sales	€ -5.091	€ -3.277	€ -3.299	€ -3.682	€ -15.349
5								
6		Gross Profit		€ 4.000	€ 3.847	€ 3.574	€ 3.012	€ 14.433
7								

Associated spreadsheet

Fig. 1. Model and associated spreadsheet ([1])

It will be possible for the user to develop a spreadsheet with the visual language in one part of the screen and seeing the associated spreadsheet in another part. The link between the model and the associated spreadsheet is bidirectional. Changes made in the model are propagated to the associated spreadsheet and vice versa.

IV. Initial Results

To get a better understanding of how users work with spreadsheets and what kind of functions they use, we have analyzed the two largest available spreadsheet corpora: EUSES and Enron [6]. We found that the majority of spreadsheets are small in terms of worksheets and formulas and do not have a high degree of coupling. Also the majority of formulas in these spreadsheets is simple. This makes it less probable that the complexity of spreadsheets is causing the high degree of errors. It supports the hypothesis that it is the spreadsheet interface that is at least partly responsible for the error-proneness of spreadsheets.

We also discovered that spreadsheet users only use a small subset of the available built-in functions. In both corpora, about 70% of the spreadsheets can be created with only 15 of the (more than 300) built-in functions. This finding further supports the idea of the feasibility of a visual language. It seems to be possible to create a visual language that consists of only a limited number of constructs, but can be used to model the majority of spreadsheets.

V. Research Plan

In our analysis of the EUSES and Enron corpora, we investigated the use of functions within a single cell. Yet a calculation in a spreadsheet does not always consist of a single cell. For example, Cell C8 in Figure 2 is the end result of the calculation. However, the formula in this cell does not contain all information for the whole calculation. For that, the formula in B8 is also needed. B8 and C8 form the calculation chain for the end result. Only when both formulas are analyzed it is possible to to get a complete list of all functions and operators that are used to calculate then end result. In future research we will use data mining techniques to search for patterns of combinations of functions that are used in calculation chains. These patterns will help us to design the constructs for the visual language. Besides the way functions are combined, we will also search for common patterns in the way the data is structured and analyze if these patterns depend on the functional domain or the type of problem that is solved in the spreadsheet.

	A	B	C	D
2		Race Time Predictor		
4		Known distance	Known time	Target distance
5		10000	0:50:00	21098
7		Projected finish time	Projected pace (min / km)	
8		1:50:19	0:05:14	

=C5*(D5/B5)^1,0

'=B8/(D5/1000)

Fig. 2. The complete formula of the calculation chain of cell C8 is: (C5 * (D5 / B5)^1.06) / (D5 / 1,000)

When we have determined the ingredients for the visual language and the best methods to structure the data in a spreadsheet, we will develop a prototype of the visual language. First we will focus on generating a spreadsheet model from the specification made with the visual language. In a next phase we will research if it is possible to make the prototype bidirectional: changes made in the visual language are propagated in the spreadsheet and changes made in the spreadsheet will find their way back in the accompanying visual language code.

When the prototype is developed it will be validated by conducting case studies. Based on our experience in industry, we will select real-life business problems and ask users to solve these problems by developing a spreadsheet using the visual language. Afterwards we will interview them and ask them about their experiences. Did the visual language give them a better overview of the spreadsheet? Did it make the development of the spreadsheet easier? In addition, we will set up controlled experiments to determine if spreadsheets developed with the visual language contain fewer errors than spreadsheets that were created in the standard way.

REFERENCES

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

[2] 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.

[3] 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.

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

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

[6] 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.