Using a Visual Language to Create Better Spreadsheets

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

Felienne Hermans
Delft University of Technology
f.f.j.hermans@tudelft.nl

ABSTRACT

It is known that spreadsheets are error-prone. It is very difficult for users to get an overview of the design of the spreadsheet, and this is causing errors. Furthermore users are not always aware of the best way to structure a spreadsheet and just start modeling. To address this we will build a visual language to develop spreadsheet models. This enables users to visualize the design of their spreadsheets. A spreadsheet generator will create the spreadsheet based on the specifications made with our visual language. During this process, best practices for structuring spreadsheets are automatically incorporated. There will be a bidirectional link between the model and the associated spreadsheet.

1. INTRODUCTION

Spreadsheets are extensively used by companies. Information embedded in these spreadsheets often forms the basis for business decisions [7]. The quality of these spreadsheets impacts the quality of the decisions. It is known that spreadsheets are error-prone [9]. A poorly structured spreadsheet is often the cause of errors. Because of the nature of the spreadsheet user interface it is difficult to keep an overview of the underlying design. And without a clear design the structure of the spreadsheet gets messy. Also users do not always possess the knowledge to structure a spreadsheet properly.

In order to address this, we present a research plan to develop an alternative user interface that enables the user to design a spreadsheet using a visual language. Based on the instructions made with this language the spreadsheet is generated automatically. The transformation between the model represented in the visual language and the automatically generated spreadsheet is bidirectional. Changes in the model are propagated to the spreadsheet and vice versa. The visual language will help the user to keep an overall overview of the spreadsheet design. And even more important, because the spreadsheet is generated automatically, well-known design patterns can be incorporated to structure the spreadsheet.

2. PROBLEM DEFINITION

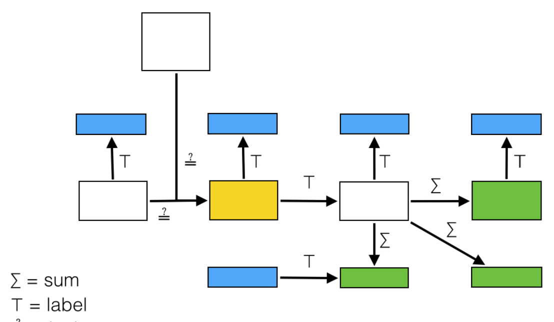
The success of spreadsheets can be partially explained by their easy-to-use interface. However, it is this same interface that is responsible for some of the problems that are associated with spreadsheets. If you write a document or a program you can scroll up and down to get an overall overview of the object you are creating. Also you can use tools (like an outline view, table of contents generator, or dependency graph) within the software to get a better overview. In Excel users can easily enter data and formulas in cells, but as soon as they hit the enter-button a spreadsheet will only show the result of the formula and not the formula itself. In a formula, references are made to other cells in the spreadsheet. It is not possible to immediately see these references. You could say the underlying design of the spreadsheet is hidden 'behind' the spreadsheet itself. The hidden design makes it difficult to understand a spreadsheet and to get an overview of the design. This is causing errors [8].

A part of the design of a spreadsheet is the way the information is structured within the sheet. There are best practices for the structure of a spreadsheet that can be found in literature. A commonly found model is to split input, model and output. This works quite well for some of the problems that people want to solve with spreadsheets. However, it is not the best model in every situation. Spreadsheets are also often used for what-if questions: "What is going to happen if I change this?" In this situation it would be better to have input and output closely together. Putting input and output closely together will lead to a completely different structure of the spreadsheet than implementation of the input, model, output principle. The optimal way to structure a spreadsheet depends on the kind of problem that you want to solve with the spreadsheet.

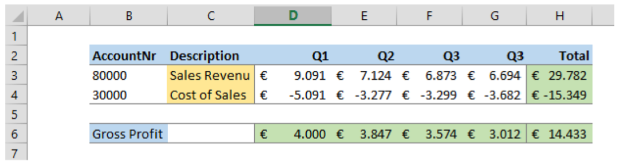
In practice, users are not making a conscious choice of how they structure their spreadsheets. They want to solve a problem as quickly as possible and just start entering the data and formulas without giving the structure a lot of thought. When the complexity of spreadsheets increases over time they end up with a messy model. At that point it is difficult to change the underlying structure and the risk of errors is imminent.

This brings us to the two problems we want to focus on in our research:

1. The design of a spreadsheet is hidden behind the spread-



Specification in 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

Figure 1: Model and associated spreadsheet

sheet itself, making it very difficult for an user to get an overview of the design

2. Users are not aware of the best way to structure a spreadsheet and just start modeling. The model works, but is poorly structured and error-prone.

3. PROPOSED SOLUTION

To address the above mentioned problems, we will develop an alternative user interface for the development of a spreadsheet. The basis for this user interface is a visual language (see also Figure 1). 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. However, if we can 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 will be possible for the user to develop a spreadsheet with the visual language in one screen and seeing the associated spreadsheet in another screen. The link between the model and the associated spreadsheet should be bidirectional. Changes made in the model should be propagated to the associated spreadsheet and vice versa.

To implement this solution we face many challenges. It it is not in the scope of this paper to address them all, but we would like to highlight two of them in more detail. First of all, we face the challenge of the scalability of a visual language. Real-life spreadsheets are often complex models and it is difficult to present such a model visually in a clear way. Careful attention should be paid to the level of details that are presented in the visual language. Second, if we want to make the link between the model en the spreadsheet bidirectional, we have to think about what kind of operations are allowed in the spreadsheet. How should we, for example, handle a change in the spreadsheet that violates the model?

4. HYPOTHESIS

There are several hypotheses underpinning the proposed solution that will be evaluated during the research:

1. Users are more likely to use a visual language than a written language as an alternative to develop a spreadsheet.
2. The representation of a spreadsheet in the visual language will help users to get an overall overview of the spreadsheet.
3. If the user has a better overall overview of the spreadsheet, the spreadsheet will contain fewer errors.
4. The automatic spreadsheet generator will use common design patterns to structure the data in a spreadsheet. This will improve the underlying design of the spreadsheets.
5. A better structured spreadsheet contains fewer errors.

5. APPROACH

A spreadsheet model consists of formulas and operations on data. These formulas and operations are the constructs of our visual language. To develop this visual language we have to research what constructs should be included. With our language it is possible to model the majority of questions that are solved with spreadsheets. This implies that we need to get an understanding of the different questions that users try to solve with spreadsheets. Finally we will also explore if the constructs we find are depending on the domain in which the spreadsheet is used or if they are used regardless of the domain.

To answer these questions we will use the EUSES spreadsheet corpus [6]. This corpus contains over 4000 real world spreadsheets from domains such as finance, biology, and education. We will analyze what kind of formulas are used in spreadsheets and how they are combined. This will be translated to the constructs that are needed to build the visual language. The spreadsheets from the EUSES corpus will be complemented with real-life spreadsheets collected from our industrial partners.

To automatically generate a spreadsheet and to structure it in the most optimal way, we will inventory the best practices in spreadsheet design. Besides, we will research if additional design patterns are needed to cover the majority of questions. We will carry out a literature study to get an overview of the commonly used and known design patterns. Furthermore, we search for additional design patterns by analyzing the spreadsheets in the EUSES corpus and the spreadsheets collected from our industrial partners.

Based on the knowledge gained about the required constructs for the visual language and the best practices to structure a spreadsheet, we develop a prototype of a 'graphical spreadsheet generator'. The spreadsheet generator will generate the spreadsheet based on the specifications made with the visual language and uses a suitable design pattern to structure the data. At this point, we also investigate if it is possible to automatically generate a graphical representation (using the syntax of the visual language) from an existing spreadsheet. This is needed to synchronize the visual model with the spreadsheet and allow the users to make changes in both the model and the spreadsheet.

6. EVALUATION

To validate our hypotheses, we carry out two different kinds of evaluations. First, we will evaluate the impact of the visual language on the behavior of the users. Is it true that they have a better overall overview of the spreadsheet and do they prefer a visual language over a written language? These two questions can be answered with case studies [11]. In the case studies we will ask users to develop a real-life spreadsheet with the new interface and afterwards interview them about their experiences.

Furthermore, we want to know if the spreadsheets that are developed with our visual language contain less errors and if this is caused by a better structure or because the end-user has a better overview/understanding of the spreadsheet or both. To evaluate this, controlled experiments will be performed. Two sets of participants are asked to develop a certain spreadsheet model. One group will use the visual language, the other group the classical spreadsheet interface. The two resulting sets of spreadsheets will be compared with each other concerning the number of errors and development time. Besides the experiments, we interview the participants to get a better insight of the users experience with the visual language.

7. RELATED WORK

Already in 2001 Burnett et al [2] developed Forms/3, a general purpose visual programming language. Main rationale to develop this language was to remove spreadsheet limitations without sacrificing consistency with the spreadsheet paradigm. Two principles in particular guided the development process: directness and immediate visual feedback.

In our approach we are less concerned with the limitations of modern spreadsheet languages. We want to improve the overall quality of spreadsheets by introducing a visual language that supports users by visualizing the design of their spreadsheet and help them to better structure their data. However, directness and especially immediate visual feedback are also two valuable guiding principles in our research.

Engels and Erwig [5] have described an automatic transformation process to generate a spreadsheet from a so called ClassSheet. The development of ClassSheets is a further elaboration of the work on spreadsheet templates [1]. With ClassSheets, it is possible to model spreadsheets according to domain-related business object structures. The ClassSheet represents both the structure and relationships of the involved (business) objects and classes and the computational details of how attributes are related and derived from each other. ClassSheets help to reduce the semantic distance between a problem domain and a spreadsheet application.

Cunha et al. [4] have further improved the concept of ClassSheets. They have embedded the ClassSheets spreadsheet models in spreadsheets themselves. Because of this, users do not have to familiarize themselves with a new programming environment. Furthermore, the authors have 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 the introduction of a visual language that does not use the tabular two-dimensional layout of spreadsheet design. We agree that users should not be required to familiarize themselves with a new programming environment before they can develop a spreadsheet. Therefore the visual language will be used in the same environment as the associated spreadsheet. However, if the model is represented in a spreadsheet-like layout, it is still difficult for users to get a good overview of the design of the model. That is the reason why we develop a language that specifies and visualizes the model at the same time.

Also, our visual language embraces object-oriented principles, but does not expect the users to be aware of these principles. The overall goal of our research is to apply software engineering principles to the design of spreadsheets to improve the overall quality of spreadsheets. However, the spreadsheet user - who is not a professional programmer - should be able to develop the spreadsheet without being required to have knowledge of these principles.

Furthermore, our visual language can be used to generate the associated spreadsheet. This does not imply that the user is restricted in influencing the layout of this spreadsheet (as is the case with the ClassSheet approach). It was estimated that 95% of U.S. Firms uses spreadsheets for financial reporting [10] and layout is an important factor for effective reporting.

Finally, the current ClassSheets approach enables the co-evolution of the spreadsheet model and the spreadsheet data. At the theoretical level, the evolution of the instance of the model and the co-evolution of the model itself has been realized [3]. In our research, we focus on bidirectional transformations and integrate them in the prototype of the spreadsheet generator.

8. EXPECTED CONTRIBUTION

This research will lead to the following contributions:

1. A classification of the type of questions that end-users try to solve with spreadsheets.
2. Better understanding of the kind of formulas that are used in spreadsheets and the way these formulas are combined to solve questions.
3. Best practices for the design/structure of spreadsheets.
4. A visual language to model spreadsheets.
5. Methods to automatically generate a spreadsheet from the visual language.
6. Methods to automatically generate a graphical representation (using the syntax of the visual language) of a spreadsheet.

7. A prototype of an alternative user interface for the development of spreadsheets that is based on a visual language.

9. REFERENCES

[1] 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*, pages 189–196. IEEE, 2005.

[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*, 11(2):155–206, 2001.

[3] J. Cunha, J. P. Fernandes, J. Mendes, H. Pacheco, and J. Saraiva. Bidirectional transformation of model-driven spreadsheets. In *Theory and Practice of Model Transformations*, pages 105–120. Springer, 2012.

[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*, pages 179–186. IEEE, 2011.

[5] 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*, pages 124–133. ACM, 2005.

[6] M. Fisher and G. Rothermel. The euses spreadsheet corpus: a shared resource for supporting experimentation with spreadsheet dependability mechanisms. *ACM SIGSOFT Software Engineering Notes*, 30(4):1–5, 2005.

[7] 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*, pages 451–460. ACM, 2011.

[8] F. Hermans, B. Sedee, M. Pinzger, and A. v. Deursen. Data clone detection and visualization in spreadsheets. In *Proceedings of the 2013 International Conference on Software Engineering*, pages 292–301. IEEE Press, 2013.

[9] R. R. Panko. What we know about spreadsheet errors. *Journal of Organizational and End User Computing (JOEUC)*, 10(2):15–21, 1998.

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

[11] R. K. Yin. *Case study research: Design and methods*, volume 5. sage, 2009.