MDSheet – Model-Driven Spreadsheets

Jácome Cunha João Paulo Fernandes Jorge Mendes
Rui Pereira João Saraiva
{jacome,jpaulo,jorgemendes,ruipereira,jas}@di.uminho.pt
HASLab/INESC TEC & Universidade do Minho, Portugal
CIICESI, ESTGF, Instituto Politécnico do Porto, Portugal
RELEASE, Universidade da Beira Interior, Portugal

ABSTRACT

This paper showcases *MDSheet*, a framework aimed at improving the engineering of spreadsheets. This framework is model-driven, and has been fully integrated under a spreadsheet system. Also, its practical interest has been demonstrated by several empirical studies.

Categories and Subject Descriptors

H.4 [**Information Systems Applications**]: Spreadsheets; D.2.0 [**Software Engineering**]: General; D.2.6 [**Software Engineering**]: Programming Environments—*Graphical environments, Integrated environments, Interactive environments*

General Terms

Languages, Design, Human Factors

Keywords

Model-Driven Spreadsheets, MDSheet, Model Inference, Embedding, Bidirectional Synchronization, Querying

1. INTRODUCTION

We can not run the modern world without spreadsheets. Spreadsheets are omnipresent, from individuals needing to cope with simple needs to large companies needing to implement complex forecasts or to produce advanced reports.

The realization of such importance has made concrete impact in the scientific community as well. This is due to more research teams devoting their efforts to improving spreadsheets, and a growing number of scientific events dedicated to them.

A successful approach to address spreadsheets under a scientific perspective consists of incorporating well-established software engineering techniques in the spreadsheet development process.

Our approach is essentially based on precisely one such technique: we adopt model-driven spreadsheet engineering. In the setting we propose a spreadsheet is abstracted through a concise model, which is then used to improve effectiveness and efficiency of spreadsheet users. The framework we describe in this paper has been realized in a traditional spreadsheet development system, thus not forcing spreadsheet users to move to a different paradigm.

The spreadsheet development framework that we envision has been fully incorporated in a tool, *MDSheet*[1], whose features include:[2]

1) Model inference: we extract the abstract representation from legacy spreadsheets;
2) Embedded models: this abstract representation is manipulated and evolved in spreadsheets themselves;
3) User guidance: relying on this business model, we are able of guiding users in avoiding traditional spreadsheet mistakes;
4) Model/instance synchronization: we support the evolution of model and instances, ensuring an automatic synchronization of the unevolved artifact;
5) Model quality assessment: a set of metrics on the complexity of a spreadsheet model can be computed;
6) Querying: spreadsheet data can be queried.

2. SPREADSHEET ENGINEERING

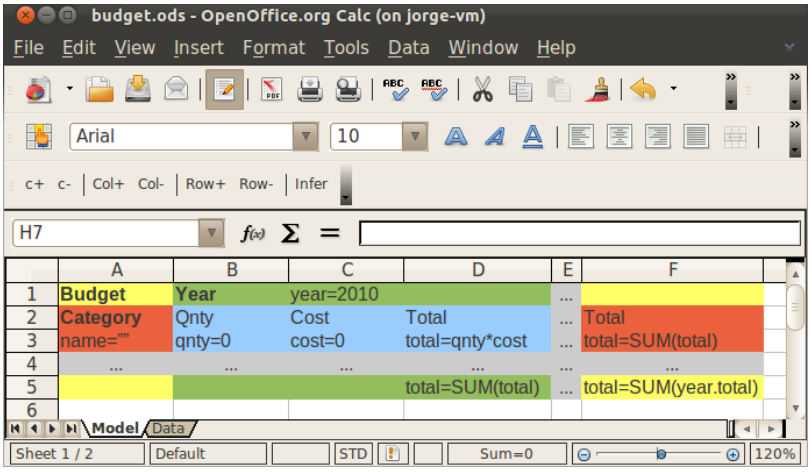
MDSheet is a framework for the engineering of spreadsheets in a model-driven fashion. This framework is highly extensible: we have actually extended it with several new functionalities that we have developed in the last few years.

2.1 Motivational Example

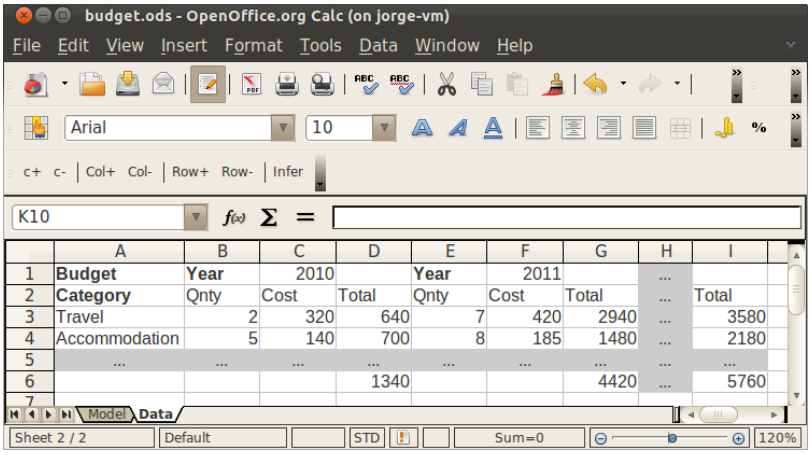
The realization of our approach to spreadsheet engineering builds upon the embedding of ClassSheets in a spreadsheet system. So, we start by introducing ClassSheets within *MDSheet* with the example given in Figure 1: we present a model for a **Budget** spreadsheet (Figure 1a), which we adapted from [13][3], and an instance of such model (Figure 1b).

This model holds three classes where data is to be inserted by end users: *i)* **Year**, with a default value of 2010, for the budget to accommodate multi-year information, *ii)* **Category**, for assigning a label to each expense and *iii)*, a(n implicit) relationship class where quantity and costs are registered and totals are calculated based on them. The actual spreadsheet may hold several repetitions of any of these elements, as indicated by the ellipsis. For each expense we record its quantity and its cost (with 0 as the default value), and we calculate the total amount associated with it. Finally, (simple) summation formulas are used to calculate the global amount spent per year (cell D5), the amount spent per expense type in all years (cell F3) and the total amount spent in all years (cell F5) are also calculated.

[1] *MDSheet* is available through the SSaaPP project website: `http://ssaapp.di.uminho.pt`.
[2] In the next section, we describe each such feature in a different subsection.
[3] We assume colors are visible in the digital version of this paper.



(a) Model worksheet.



(b) Data/instance worksheet.

Figure 1: A bidirectional model-driven environment for a budget spreadsheet.

Following is the description of the full set of features offered by *MDSheet*.

2.2 Model Inference

A model-driven approach to spreadsheet engineering offers an improved development experience: an abstract representation of a spreadsheet, i.e., its model, helps us, among other things, in guiding users into preventing errors. This approach, however, requires the definition of a model in parallel with the spreadsheet it abstracts. In order to handle legacy spreadsheets, i.e., the ones that have not been developed from scratch together with their model, we have devised a model inference technique [2], that has been implemented in *MDSheet*. Concretely, we infer models in the ClassSheets language, an object-oriented high-level formalism to abstract spreadsheets [13].

2.3 Embedded Models

The worksheet structure of spreadsheets is a decisive factor in their modularity. In fact, we exploited precisely this structure to make the model of a spreadsheet available within spreadsheet systems themselves: one worksheet holds the model of a spreadsheet, while another holds its data. This embedding of spreadsheets has also been implemented under *MDSheet* [6], which was demonstrated in Section 2.1. Moreover, we extended the ClassSheets language with database constraints, such as unique columns/rows or foreign keys, which have also been incorporated in *MDSheet* [11]. In fact, we have further extended the available restrictions so that the user can specify the contents of a cell using regular expressions or intervals [8]. Finally, we extended ClassSheets with references between different models making them more flexible. Note that through this embedding we can guarantee that spreadsheet data always conforms to a model.

2.4 User Guidance

The embedding of our extended version of the ClassSheet language allows us to guide the user in inserting correct data. When a model is designed, it serves as a guider in the creation of a data worksheet, which is initially empty. Only cells containing plain data can be edited as all other are inferred from the model. This prevents, e.g., users from making mistakes when defining formulas as they are locked. Moreover, the restrictions created in the model guarantee that the data in the cells respects them. In the model it is possible to define an interval of integers for a cell, or a regu-

lar expression that the content must conform to. A column or row can be marked as having only unique values or being a foreign key to another column or row. All these restrictions are enforced by *MDSheet*. In the case of foreign keys, the user can use a combo box to select existing values from the referred column/row.

2.5 Model/Instance Synchronization

As any other software artifact, spreadsheets evolve over time. *MDSheet* accommodates changes by allowing the evolution of models and instances, while automatically coevolving the unchanged artifact. For this, we introduced a formal framework to allow evolutions of the model to be automatically spanned to the instances [6, 7, 12]. We have later proposed techniques and tools to the evolution of data by the user and corresponding automatic coevolution of the model [3]. We therefore ensure that model/instance consistency is never broken.

2.6 Model Quality Assessment

In a first attempt to measure the quality of a spreadsheet model, we introduced a set of metrics to calculate the complexity of ClassSheet models [9]. These metrics are implemented under *MDSheet* and can be calculated for any ClassSheet defined using it. They are then compared to the same metrics computed for a repository of ClassSheet models so users can have a reference point for such values. The evolution mechanisms can then be used to evolve the spreadsheet improving it according to the metrics calculated.

2.7 Querying

As many spreadsheets are used as data repositories, the need to query their data is frequent. *MDSheet* also integrates a query system, which allows the definition of model-oriented queries, in the style of traditional database queries. This allows the writing of queries without having to manually observe a possibly large number of columns and rows of concrete data. Indeed, queries are written, by analyzing models, as abstractions that are simpler to understand. Our system was initially presented as a textual language [1, 4], very similar to SQL. Even being textual it already was of great help for users [14]. Still, we have further improved it by embedding the language in a worksheet, thus creating a visual language for spreadsheet querying [5].

3. EMPIRICAL VALIDATION

One of the purposes of our tool is to help users commit less errors; if possible, it also intends to help users work faster with spreadsheets. To assess these two concerns we have run an empirical study and we have found empirical evidence that indeed our model-driven spreadsheet environment can in fact help users become more efficient and more effective [10].

4. CONCLUSION

We briefly presented *MDSheet* and all the features it offers to its users. Given the fact that it has been built as a framework, new tools, even if not proposed by us, can easily be integrated in it.

We believe this tool is in a very mature state and can be used in real case scenarios. We have thus started its integration in industry: i), to support test case evolution in an agile testing framework of a software house; ii), to adapt data produced by different database systems for a car multimedia production company; and iii), to provide spreadsheet models for a food bank.

Acknowledgments

This work is part funded by the ERDF - European Regional Development Fund through the COMPETE Programme (operational programme for competitiveness) and by National Funds through the FCT - Fundação para a Ciência e a Tecnologia (Portuguese Foundation for Science and Technology) within project `FCOMP-01-0124-FEDER-020532`. The first author was funded by the FCT grant `SFRH/BPD/73358/2010`.

5. REFERENCES

[1] O. Belo, J. Cunha, J. P. Fernandes, J. Mendes, R. Pereira, and J. Saraiva. Querysheet: A bidirectional query environment for model-driven spreadsheets. In *VLHCC '13*, pages 199–200. IEEE CS, 2013.

[2] J. Cunha, M. Erwig, and J. Saraiva. Automatically inferring classsheet models from spreadsheets. In *VLHCC '10*, pages 93–100. IEEE CS, 2010.

[3] J. Cunha, J. P. Fernandes, J. Mendes, H. Pacheco, and J. Saraiva. Bidirectional transformation of model-driven spreadsheets. In *ICMT '12*, volume 7307 of *LNCS*, pages 105–120. Springer, 2012.

[4] J. Cunha, J. P. Fernandes, J. Mendes, R. Pereira, and J. Saraiva. Querying model-driven spreadsheets. In *VLHCC '13*, pages 83–86. IEEE CS, 2013.

[5] J. Cunha, J. P. Fernandes, J. Mendes, R. Pereira, and J. Saraiva. Embedding model-driven spreadsheet queries in spreadsheet systems. In *VLHCC '14*, 2014. to appear.

[6] J. Cunha, J. P. Fernandes, J. Mendes, and J. Saraiva. Embedding and evolution of spreadsheet models in spreadsheet systems. In *VLHCC '11*, pages 186–201.

[7] J. Cunha, J. P. Fernandes, J. Mendes, and J. Saraiva. MDSheet: A framework for model-driven spreadsheet engineering. In *ICSE 2012*, pages 1412–1415. ACM.

[8] J. Cunha, J. P. Fernandes, J. Mendes, and J. Saraiva. Extension and implementation of classsheet models. In *VLHCC '12*, pages 19–22. IEEE CS, 2012.

[9] J. Cunha, J. P. Fernandes, J. Mendes, and J. Saraiva. Complexity Metrics for ClassSheet Models. In *ICCSA '13*, volume 7972, pages 459–474. LNCS, 2013.

[10] J. Cunha, J. P. Fernandes, J. Mendes, and J. Saraiva. Embedding, evolution, and validation of spreadsheet models in spreadsheet systems. 2014. submitted.

[11] J. Cunha, J. P. Fernandes, and J. Saraiva. From Relational ClassSheets to UML+OCL. In *SAC '12*, pages 1151–1158. ACM, 2012.

[12] J. Cunha, J. Visser, T. Alves, and J. Saraiva. Type-safe evolution of spreadsheets. In D. Giannakopoulou and F. Orejas, editors, *FASE '11*, volume 6603 of *LNCS*, pages 186–201. Springer, 2011.

[13] G. Engels and M. Erwig. ClassSheets: automatic generation of spreadsheet applications from object-oriented specifications. In *ASE '05*, pages 124–133. ACM, 2005.

[14] R. Pereira. Querying for model-driven spreadsheets. Master's thesis, University of Minho, 2013.