Model-Driven Spreadsheets in a Multi-User Environment

Jorge Mendes
HASLab / INESC TEC & Universidade do Minho, Portugal
jorgemendes@di.uminho.pt

I. Introduction

Spreadsheets are widely used by non-professional programmers, the so-called end-users, to perform simple calculations, but also by professional programmers in large software organizations, where spreadsheets are used to collect information from different systems, to transform data coming from one system to the format required by another, or to present data in human-friendly form.

The tremendous success of spreadsheets has been driven by the simplicity in their usage, the high degree of flexibility they provide, and ultimately by the power of the visual interactive environment provided by spreadsheet systems.

Being artifacts that usually experience long lives, spreadsheets and their evolution pose an interesting challenge similar to the one of evolving other software languages. Usually, spreadsheets are created by single end-users, without planning ahead of time for maintainability or scalability. Over time, however, spreadsheets become increasingly complex software systems: they are used to process increasing amounts of data and supporting increasing numbers of end-users. In fact, it is common to find spreadsheets with a (too) large number of cells/columns/rows, with intricate dependencies between worksheets that make the comprehension of the underlying business logic extremely difficult!

One of the promising research directions that attempts to tackle this problem proposes the use of a model-driven approach to spreadsheet engineering. This approach finds its inspiration in other engineering disciplines, such as the civil engineering, that is using models for centuries.

In the context of spreadsheet engineering, we realize models as elements within the *ClassSheets* language [1], and these are used to realize the business model of a spreadsheet. This language has later been integrated in a general framework for model-driven spreadsheet evolution [2]–[5]: models and instances are created and evolved under the same environment, and changes in either artifact are not only possible but also automatically reflected on the other.

This approach, however, considers an environment where only one instance is possible for each model. This means that the realistic scenario where several users and instances of the same model co-exist has still not yet been considered. And this is indeed a scenario that one often finds, e.g., within the context of most companies.

The main goal of our work is to develop a framework for model-driven and distributed collaboration in complex spreadsheet ecosystems. Indeed, in a computing world that is growingly distributed, the process of elaborating spreadsheets is often performed in a collaborative way, by many actors. In a model-driven environment, this means that one model must have many instances, one for each of the actors involved. Later, instances may be evolved independently and synchronization techniques are needed to evolve the original model so that conformity is restored.

II. Model-Driven Spreadsheet Engineering

In the past, model-driven approaches have been proposed to address problems that are often encountered in spreadsheets. This approach, for example, prevents faulty formulas from occurring since these are automatically inserted according to the information on the model. We have also pursued one such approach, that we describe in the remaining of this section.

We have started by embedding *ClassSheet* models in a spreadsheet system [3], providing a coherent environment for model-driven spreadsheet development. This allows us to develop both the model and its conforming data in the same environment. Also, this is precisely the environment that spreadsheet users are accustomed to, and further allows the evolution of a model while the co-related data instance is automatically co-evolved.

Later, and in order to provide a more powerful environment, we have improved our system to allow structural evolution steps in the data while it is the model that is automatically co-evolved [4].

By then, we had developed a bidirectional model-driven technique for spreadsheet engineering which was integrated in a fully-functional system, by means of an addon [5] built in a widely used spreadsheet system.

Furthermore, we have extended *ClassSheet* models to integrate in them several type information for attributes [6], [7], so that we can restrict even further the occurrence of incorrect values in spreadsheet data.

Unfortunately, all our previous works suffer from a severe limitation: they were conceived to allow a single data instance per model. This is a setup that is not realistic for most organizations, where several, possibly distributed, instances are accessed and modified by several users at the same time.

To address this concern, we propose to develop a multi-user environment that is able to support several instances of the same model. We want to allow those instances to be evolved

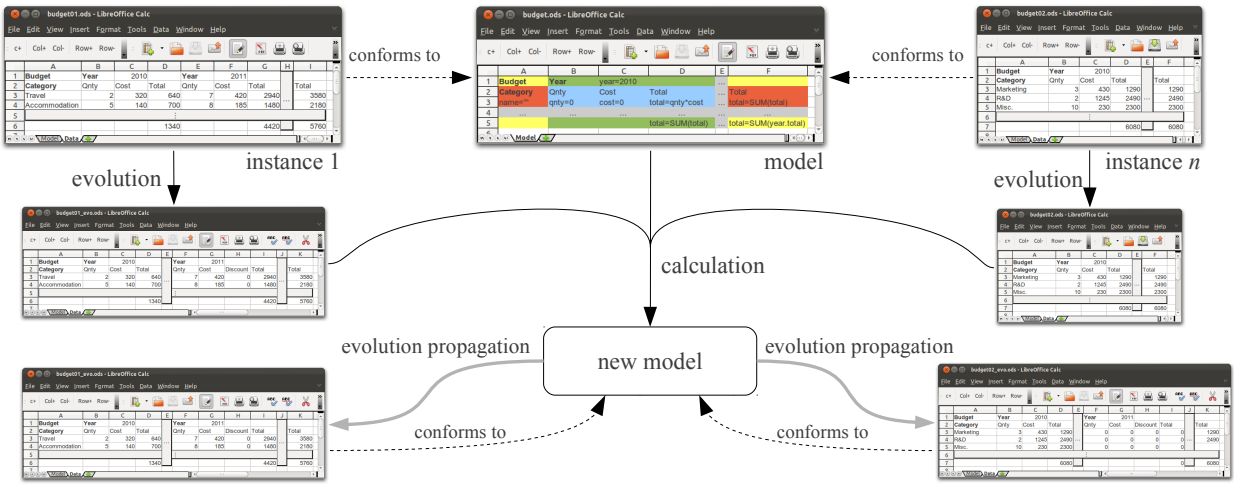

Figure 1. In the environment to develop, we want to allow several instances of the same model. Each instance may then be evolved independently, and the conformity relationship of the model with all its instances needs to be restored. This involves calculating a new model and eventually propagation changes back to all the instances.

independently and at any time, while their synchronization with the underlying model is always guaranteed.

III. Multi-User Spreadsheet Environment

In a distributed evolution setting, model-driven spreadsheet engineering raises a number of interesting research questions:

- Will all the evolved spreadsheets conform to the original *ClassSheet* model?
- If not, how can we update the model in such a way that conformity is restored?
- When several models are possible, what strategies lead to the best model being obtained?

An approach to solve this problem is to find a new model consistent with the new data sheets, but this poses other problems since there can be no model consistent with all the instances. Refining this idea, we can find a model that can store all the data and computations of the instances, and then perform some evolution steps on the data to make them conform to the new model. We foresee that this can be broken down in the following steps:

a. Define an algebra of *ClassSheets*: Synchronizing the same model with multiple instances requires being able to abstractly reason about *ClassSheet* models. We must be able to: test whether two models define equivalent spreadsheets; merge two models into a single one; realize the minimal set of changes to perform in a model to restore conformity. Given that the repetition structure of *ClassSheet* resembles the one found in regular expressions and also that the *ClassSheet* language is regular, we plan to explore techniques from automata theory.

b. Study the evolution of distributed spreadsheets: Once *ClassSheets* are elements of an algebra, we will study techniques for the evolution of both models and instances in the presence of several instances, possibly distributed by different computers or in a cloud-based environment. For this, we will combine model comparison and bidirectional transformation techniques.

c. Empirical validation: The effective maintenance of software over time is critical. In addition to the problem of evolving software (i.e., spreadsheets), we need to guarantee that models are still manageable after several model and instance evolution steps. This can be assessed with a set of empirical studies with real users that can highlight opportunities to improve the employed techniques.

IV. Conclusion

We have created a bidirectional model-driven spreadsheet environment. In our environment, however, only one instance can be defined per model. Now, we propose to improve this environment so that several instances of one model can be edited and evolved at the same time, whilst never losing the synchronization of the entire system.

References

[1] G. Engels and M. Erwig, "ClassSheets: automatic generation of spreadsheet applications from object-oriented specifications," in *ASE'05: Proc. of the 20th IEEE/ACM Int. Conf. on Automated Software Engineering*. ACM, 2005, pp. 124–133.

[2] J. Cunha, J. Visser, T. Alves, and J. Saraiva, "Type-safe evolution of spreadsheets," in *FASE'11/ETAPS'11: Proc. of the 14th Int. Conf. on Fundamental Approaches to Software Engineering*. Springer-Verlag, 2011, pp. 186–201.

[3] J. Cunha, J. Mendes, J. P. Fernandes, and J. Saraiva, "Embedding and Evolution of Spreadsheet Models in Spreadsheet Systems," in *VL/HCC'11: IEEE Symp. on Visual Languages and Human-Centric Computing*. IEEE Computer Society, 2011, pp. 186–201.

[4] J. Cunha, J. P. Fernandes, J. Mendes, H. Pacheco, and J. Saraiva, "Bidirectional Transformation of Model-Driven Spreadsheets," in *ICMT'12: 5th Int. Conf. on Model Transformation*, 2012, pp. 105–120.

[5] J. Cunha, J. P. Fernandes, J. Mendes, and J. Saraiva, "MDSheet: A Framework for Model-driven Spreadsheet Engineering," in *ICSE'12: Proc. of the 34rd Int. Conf. Software Engineering*. ACM, 2012, pp. 1395–1398.

[6] J. Cunha, J. P. Fernandes, and J. Saraiva, "From Relational ClassSheets to UML+OCL," in *SAC'12: the Software Engineering Track at the ACM Symposium On Applied Computing*. ACM, 2012, pp. 1151–1158.

[7] J. Cunha, J. P. Fernandes, J. Mendes, and J. Saraiva, "Extension and Implementation of ClassSheet Models," in *VL/HCC'12: IEEE Symp. on Visual Languages and Human-Centric Computing*. IEEE Computer Society, 2012, to appear.