Towards Systematic Spreadsheet Construction Processes

Jorge Mendes*, Jácome Cunha†, Francisco Duarte§, Gregor Engels‡, João Saraiva* and Stefan Sauer‡
*HASLab, INESC TEC & Universidade do Minho, Portugal, email: {jorgemendes,saraiva}@di.uminho.pt
†NOVA LINCS, DI, FCT, Universidade NOVA de Lisboa, Portugal, email: jacome@fct.unl.pt
§Bosch Car Multimedia, Portugal, email: Francisco.Duarte@pt.bosch.com
‡Universität Paderborn, Germany, email: {engels,sauer}@upb.de

***Abstract*—Spreadsheets are used in professional business contexts to make decisions based on collected data. Usually, these spreadsheets are developed by end users (e.g. accountants) in an ad-hoc way. The effect of this practice is that the business logic of a concrete spreadsheet is not explicit to them. Thus, its correctness is hard to assess and users have to trust.**

We present an approach where structure and computational behavior of a spreadsheet are specified by a model with a process-like notation based on combining pre-defined functional spreadsheet services with typed interfaces. This allows for a consistent construction of a spreadsheet by defining its structure and computational behavior as well as filling it with data and executing the defined computational behavior. Thus, concrete spreadsheets are equipped with a specification of their construction process. This supports their understanding and correct usage, even in case of legacy spreadsheets.

The approach has been developed in cooperation with an industrial partner from the automotive industry.

***Keywords*-model-driven engineering; situational method engineering; construction process; spreadsheet**

I. Introduction

Spreadsheets are used in professional business contexts to make calculations and decisions based on collected data: it is estimated that 95% of all U.S. organizations use spreadsheets for financial reporting [15], 90% of all analysts in industry perform calculations in spreadsheets [15], and 50% of all spreadsheets are the basis for decisions [13]. Spreadsheets are not only used to define worksheets containing data and formulas, but also to collect information from different systems, to adapt data produced from one system to the format required by another, to perform operations to enrich/simplify data, to present data in a graphical (visual) representation, etc.

It has been observed that, despite admitting serious risks, many organizations manipulate large spreadsheets in a frightening ad-hoc way: they are adapted/enriched/evolved by using an unspecified/undocumented process, usually performed by users directly updating the computational structure or data of a spreadsheet. This is even more frightening as the business logic of a concrete spreadsheet is hidden, baffling and difficult to understand by end users. Thus, the correctness of a spreadsheet is hard to assess and users have to trust.

This work is financed by the ERDF – European Regional Development Fund through the Operational Programme for Competitiveness and Internationalisation - COMPETE 2020 Programme within project POCI-01-0145-FEDER-016718, and by FCT as part of project UID/CEC/04516/2013. This work is also financed by the bilateral project FCT/DAAD with ref. 441.00. The first author is funded by FCT grant SFRH/BD/112651/2015.

A careful analysis of this situation reveals a few shortcomings in the ad-hoc construction of spreadsheet applications:

- Spreadsheets do not have the notion of a type level. Thus, spreadsheets are developed on the instance level where no type checking is possible. This leads quite often to structural changes, like insertion of rows and columns, that are not complete and consistent as certain dependencies are only implicit and cannot be detected by the spreadsheet system. This situation occurred in the spreadsheet used for an economical study on the level of austerity that countries should comply with [2].
- The missing concept of types can lead to inconsistent computation sequences where the data flow between two computation steps is not appropriate [12].
- The internal data and control flow of computation steps within a spreadsheet is only implicitly defined by a spreadsheet developer. The missing explicitness and documentation hinders any kind of development traceability and understandability of a given, often legacy, spreadsheet as well as any kind of maintenance [12].

We combine several approaches from the literature to propose a novel solution which enables precise, understandable and repeatable construction of spreadsheets.

First, ClassSheets which have been previously introduced [10], [5], [6], [7], allow developers to define the logic and structure of a spreadsheet on the type level. However, so far only the specific part of the spreadsheet cells and formulas has been targeted, leaving out several features commonly used in industrial situations (e.g. pivot tables).

Second, Model-Driven Engineering (MDE) has been introduced as a methodology to specify structural and behavioral aspects of a system on an abstract model level, before it is implemented [14]. MDE is also applied for process-modeling tasks, e.g. in business or production process modeling. Having a fully-fledged process allows to enact it and to automate the execution of individual steps of the process. In fact, MDE has been previously adapted for spreadsheet [4], [9], [3], [8], but again only to address the issues related to cells.

Third, Situational Method Engineering is an approach to define processes for a certain development task by composing predefined parameterized building blocks (i.e., method services) with typed interfaces to a consistent process [11].

The combination and adoption of these three approaches yields our novel and integrated approach to define the *construction process of spreadsheets* for business applications. This construction process comprises (a) defining structure and computational behavior of a spreadsheet (construction process design) as well as (b) filling it with data and executing the defined computational behavior (construction process enactement). The basic idea of our approach is to equip a spreadsheet with an operational specification based on *functional spreadsheet services*. This yields consistent spreadsheets with a documented internal computation structure.

The proposed framework as been developed in collaboration with an industrial partner, Bosch Car Multimedia Portugal, where we endeavored a large and complex case study.

II. Spreadsheet Construction Framework

The framework we propose aims to provide a safe way to design and construct spreadsheets. The key concepts behind it are the specification of the actions that are to be undertaken to create and use a spreadsheet, which we call *functional services*, and the *artifacts* and their *types* that are required by or originate from these actions.

The functional services referred in this work are the ones that users have available in common spreadsheet systems (e.g. insert a pivot table). This work provides a way to specify existing actions, but also new features that may be introduced in the future in spreadsheet systems, by proving generic types to define them. The functional services are described as activities that can receive and produce artifacts as input and output.

The artifacts are typed, that is, we make a distinction between any two artifacts that are not compatible with each other or that do not serve the same purpose. For instance, a *chart* is a different type than a *pivot table*. This allows us to restrict the inputs of the actions to be performed since they usually only work on specific artifacts.

The actions and artifacts are combined in an UML activity diagram [1], indicating the control and data flows of the spreadsheet construction process.

The process of creating and executing a construction process for spreadsheets is organized in three distinct phases as illustrated in Figure 1:

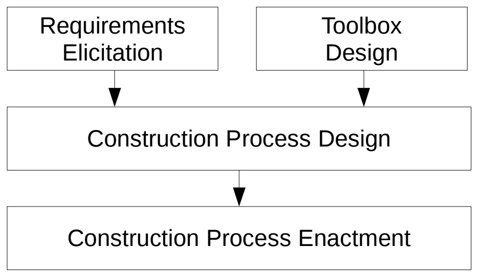

Fig. 1. The three phases of creating and executing the spreadsheet construction process.

In the first phase, two distinct and parallel tasks are performed. On the one hand, requirements elicitation for the spreadsheet application is executed. Since this is a task well studied, we will not discuss it further. On the other hand, a toolbox of reusable, parameterizable, and typed functional spreadsheet services (e.g. insert a pivot table, insert a chart) has to be provided by experts in information/spreadsheet systems. In this task the functional services are identified as well as their input and output (types). We provide a first version of the toolbox which can later be extended. The toolbox also supports data types which typically occur in a spreadsheet (e.g. pivot table, chart).

In the next step – construction process design – the functional spreadsheet services from the toolbox are instantiated and integrated in the construction process. The types available in the toolbox are used in interface definitions of the parameterized functional services. These building blocks are composed by a spreadsheet designer during the construction process design. As we are focusing here on professional business spreadsheets, we assume that this role of a spreadsheet designer exists. It might be assigned to an experienced end user or a dedicated spreadsheet expert.

Finally, the construction process is (automatically) enacted by the spreadsheet user, which means that a spreadsheet is created, filled with data and its functional services executed. This enactment of the construction process can be repeated by an end user regularly with e.g. new data. Such procedure is typical when spreadsheets are used for business applications which are regularly, e.g. daily, weekly, or monthly, reused for certain monitoring or controlling tasks.

The proposed framework has been developed in collaboration with an industrial partner: the Bosch Car Multimedia. We have been conducting a large and complex case study: their quality report spreadsheet-based system. In this system large amounts of data are collected from two SAP systems. The data is then manually transformed by Bosch engineers so that a monthly report is delivered to the administration in Germany. In our study, we have shown how such an ad-hoc, error-prone and time-consuming task is documented/specified and automated using our approach.

III. Conclusion

In this paper we present a systematic construction process to make the computation flow within a spreadsheet explicit. This process is composed of predefined typed functional services, thus making the structure and internal computation flow of spreadsheets observable. This process also allows to automate the construction of spreadsheets and to ease auditing.

The proposed approach originated from ideas gathered within an industrial case at Bosch Car Multimedia, Portugal. A prototype is being developed to evaluate the approach with this industrial case. The evaluation will further be enlarged to other case studies and to re-engineering existing spreadsheets used in commercial and industrial applications. This will be done in close cooperation with industrial partners from the Software Innovation Campus Paderborn (SICP).

REFERENCES

[1] J. Arlow and I. Neustadt. *UML 2.0 and the Unified Process: Practical Object-Oriented Analysis and Design (2Nd Edition)*. Addison-Wesley Professional, 2005.

[2] P. Coy. Faq: Reinhart, rogoff, and the excel error that changed history. Bloomberg: http://www.bloomberg.com/news/articles/2013-04-18/faq-reinhart-rogoff-and-the-excel-error-that-changed-history, 2013 April.

[3] J. Cunha. *Model-Based Spreadsheet Engineering*. PhD thesis, University of Minho, March 2011.

[4] J. Cunha, J. P. Fernandes, J. Mendes, H. Pacheco, and J. Saraiva. Bidirectional transformation of model-driven spreadsheets. In Z. Hu and J. de Lara, editors, *Theory and Practice of Model Transformations*, volume 7307 of *Lecture Notes in Computer Science*, pages 105–120. Springer, 2012.

[5] J. Cunha, J. P. Fernandes, J. Mendes, R. Pereira, and J. Saraiva. Embedding model-driven spreadsheet queries in spreadsheet systems. In *Proceedings of the 2014 IEEE Symposium on Visual Languages and Human-Centric Computing*, VL/HCC '14, pages 151–154, Washington, DC, USA, 2014. IEEE Computer Society, IEEE Computer Society.

[6] J. Cunha, J. P. Fernandes, J. Mendes, and J. Saraiva. MDSheet: A Framework for Model-driven Spreadsheet Engineering. In *Proceedings of the 34rd International Conference on Software Engineering*, ICSE'12, pages 1395–1398. ACM, 2012.

[7] J. Cunha, J. P. Fernandes, J. Mendes, and J. Saraiva. Embedding, evolution, and validation of model-driven spreadsheets. *IEEE Transactions on Software Engineering*, 41(3):241–263, March 2015.

[8] J. Cunha, J. P. Fernandes, J. Mendes, and J. Saraiva. Spreadsheet engineering. In V. Zsók, Z. Horváth, and L. Csató, editors, *Central European Functional Programming School: 5th Summer School, CEFP 2013, Cluj-Napoca, Romania, July 8-20, 2013, Revised Selected Papers*, pages 246–299. Springer International Publishing, Cham, 2015.

[9] J. Cunha, J. Mendes, J. Saraiva, and J. Visser. Model-based programming environments for spreadsheets. *Science of Computer Programming*, 96, Part 2:254–275, 2014.

[10] 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*, ASE '05, pages 124–133. ACM, 2005.

[11] B. Henderson-Sellers, J. Ralyté, P. J. Ågerfalk, and M. Rossi. *Situational Method Engineering*. Springer, 2014.

[12] F. Hermans. *Analyzing and visualizing spreadsheets*. PhD thesis, Delft University of Technology, 2012.

[13] 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*, ICSE'11, pages 451–460, New York, NY, USA, 2011. ACM.

[14] S. Kent. *Model Driven Engineering*, pages 286–298. Springer Berlin Heidelberg, Berlin, Heidelberg, 2002.

[15] R. R. Panko and N. Ordway. Sarbanes-oxley: What about all the spreadsheets? *CoRR*, abs/0804.0797, 2008.