Journal of Visual Languages and Computing 23 (2012) 267–286

Contents lists available at SciVerse ScienceDirect

Journal of Visual Languages and Computing

journal homepage: www.elsevier.com/locate/jvlc

Systematic evolution of model-based spreadsheet applications ☆

Markus Luckey [a,*], Martin Erwig [b], Gregor Engels [a]

[a] University of Paderborn, 33098 Paderborn, Germany
[b] Oregon State University, Corvallis, OR 97331-3202, USA

ARTICLE INFO

Article history:
Received 2 August 2010
Accepted 1 November 2011
Available online 9 June 2012

Keywords:
Model-based
Spreadsheet
Evolution
Update
Propagation

ABSTRACT

Using spreadsheets is the preferred method to calculate, display or store anything that fits into a table-like structure. They are often used by end users to create applications, although they have one critical drawback—spreadsheets are very error-prone. Recent research has developed methods to reduce this error-proneness by introducing a new way of object-oriented modeling of spreadsheets before using them. These spreadsheet models, termed ClassSheets, are used to generate concrete spreadsheets on the instance level. By this approach sources of errors are reduced and spreadsheet applications become easier to understand.

As usual for almost every other application, requirements on spreadsheets change due to the changing environment. Thus, the problem of evolution of spreadsheets arises. The update and evolution of spreadsheets is the uttermost source of errors that may have severe impact.

In this paper, we will introduce a model-based approach to spreadsheet evolution by propagating updates on spreadsheet models (i.e. ClassSheets) to spreadsheets. To this end, update commands for the ClassSheet layer are automatically transformed to those for the spreadsheet layer. We describe spreadsheet model update propagation using a formal framework and present an integrated tool suite that allows the easy creation and safe update of spreadsheet models. The presented approach greatly contributes to the problem of software evolution and maintenance for spreadsheets and thus avoids many errors that might have severe impacts.

1. Introduction

Spreadsheets are one of the most popular and important programming languages used in business applications today. Estimates say that "each year tens of millions of managers and professionals around the world create hundreds of millions of spreadsheets" [1]. Reasons for this wide-spread use of spreadsheets are the ease of creating highly sophisticated spreadsheet applications using the simple and intuitive two-dimensional tabular layout and of course the fast and broad availability of spreadsheet applications. Due to their availability to non-experts, spreadsheets belong to the category of end-user development environments. However, the simplicity of spreadsheets is misleading. Although, the spreadsheet user group usually is able to develop complex spreadsheets, the users often do not have the knowledge to prevent errors leading to error-prone and unstructured spreadsheets [2]. Studies estimate rates of at least 80 percent of erroneous spreadsheets [3,4]. How costly these errors can be is evident in recent news stories. For example, in 2006, the Office of Government Commerce Buying Solutions erred in informing 29 suppliers that they had been successful in the public sector tendering process. In a

☆ This paper has been recommended for acceptance by Shi Kho Chang.

* Corresponding author. Tel.: +49 5251 60 3844.
E-mail addresses: luckey@upb.de (M. Luckey), erwig@eecs.oregonstate.edu (M. Erwig), engels@upb.de (G. Engels).

1045-926X/$ - see front matter © 2012 Elsevier Ltd. All rights reserved.
http://dx.doi.org/10.1016/j.jvlc.2011.11.009

subsequent letter, they stated: "Unfortunately, [we are] not, as we had hoped, in a position to accept your tender at this time. This is because an error in the original evaluation spreadsheet has been identified, necessitating rescoring of all tenders for this project…this error has now been corrected and this has caused a small number of changes to the original award decision" [5].

Reasons for these failures are manifold, the most important reason being the missing business model. A business model of spreadsheets specifies which business entities are represented by the specific spreadsheet. For instance, a spreadsheet for budget calculation may comprise entities like *category* or *year* (cf. Fig. 1, categories in rows 4 and 5 and years in columns **C** up to **H**). A spreadsheet's business model is not given explicitly to the spreadsheet application, but usually is kept in the developer's idea and captured implicitly in the spreadsheet's layout and data. Taking into account the complexity of today's spreadsheets, the gap between the *implicit business model* of a spreadsheet and the resulting implementation (i.e. the spreadsheet itself) is too large. Fig. 2 shows the current approach of spreadsheet development. The user has an idea of the spreadsheet's business model in mind and develops the spreadsheet accordingly. However, spreadsheet development is very low-level and current spreadsheet applications do not allow to implement all elements of the business model. The semantic gap between the implicit business model and the spreadsheet leads to misleading error reports (e.g. the spreadsheet application complains about a formula that differs from surrounding formulas) or even missed errors. Spreadsheet applications like Microsoft Excel fail to sufficiently mitigate this situation. The reason for this failure is that the business model is not described explicitly and thus cannot be used to automatically validate the current state of a spreadsheet (i.e. the inserted data, formulas, and references). Bridging the described gap between the business model (captured in the developer's idea) and the IT implementation is recently known under *Business/IT Alignment*. The need for Business/IT alignment was emphasized by the Sarbanes-Oxley Compliance (see [6]) leading to plenty of tools that are concerned with increasing the quality of use and development of spreadsheets. However, these approaches rather provide process guidance and support security aspects but do not tackle the problem at its root, the gap between business and IT. But how can the business model help to prevent errors?

Usually, errors are produced while changing a spreadsheet. We distinguish two different kinds of spreadsheet changes, namely instance evolution and model evolution. Both kinds of changes are part of *spreadsheet evolution*. Instance evolution describes changes that concern a spreadsheet's data but not its representation and interrelations. For instance, inserting a new category in the budget spreadsheet mentioned above is part of instance evolution. In turn, model evolution describes changes that concern the relation between data (e.g. formulas) or the insertion and deletion of data *types*. For instance, the insertion of a new column that holds a new *type* of data (e.g. an exchange rate for given costs) is a change at the underlying business model. Those changes must be applied to all inserted data, e.g. the exchange rate must be inserted for all categories and years. See Fig. 3 for the distinction of instance evolution and model evolution. For reasons of understandability, we chose UML class diagrams [7] to represent the business model. Of course every other representation is applicable, e.g. the Entity Relationship Model [8]. Both instance and model evolutions are sources of errors. The most frequent error in

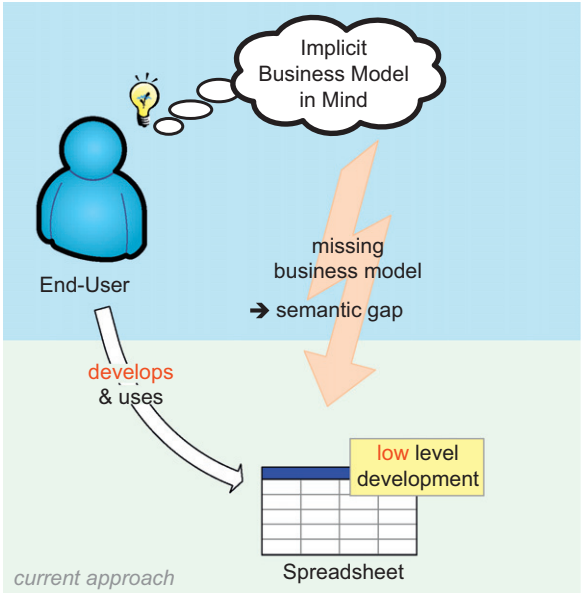

Fig. 2. The current spreadsheet development approach.

	A	B	C	D	E	F	G	H	I
1	**Budget**		**Year**			**Year**			
2			2009			2010			
3	**Category**	Name	Qnty	Cost	Total	Qnty	Cost	Total	Total
4		Apples	0	0	=(C4*D4)	0	0	=(F4*G4)	=SUM(E4;H4)
5		Bananas	0	0	=(C5*D5)	0	0	=(F5*G5)	=SUM(E5;H5)
6	Total				=SUM(E4;E5)			=SUM(H4;H5)	=SUM(E6;H6)

Fig. 1. A spreadsheet for budget calculation.