End-users Productivity in Model-based Spreadsheets: An Empirical Study*

Laura Beckwith[1], Jácome Cunha[2,3], João Paulo Fernandes[2,4], and João Saraiva[2]

[1] HCIResearcher, Denmark, beckwith@hciResearcher.com
[2] Universidade do Minho, Portugal, {jacome, jpaulo, jas}@di.uminho.pt
[3] ESTGF, Instituto Politécnico do Porto, Portugal
[4] Universidade do Porto, Portugal

Abstract. Spreadsheets are widely used and studies show that most of the existing ones contain non-trivial errors. To improve end-users productivity, recent research proposes the use of a model-driven engineering approach to spreadsheets. In this paper we conduct the first empirical study to assess the effectiveness and efficiency of this approach. A set of spreadsheet end users worked with two different model-based spreadsheets. We present and analyze here the results achieved.

1 Introduction

Spreadsheets (SS) can be viewed as programming environments for non-professional programmers, the so-called "end users" [7]. End-user programmers vastly outnumber professional ones and create hundreds of millions of spreadsheets every year [10], especially for developing business applications. As numerous studies have shown, this high production rate is accompanied by an alarming high rate of errors, with some reports claiming that 90% of real-world spreadsheets contain errors [8, 9].

In order to improve the robustness of SS, a considerable amount of research has been done [1, 3–6]. One of the promising solutions advocates the use of a *Model-Driven Engineering* (MDE) approach, in which a business model of the spreadsheet data is defined; end users are then guided to introduce data that conforms to the model [4]. Several models have been proposed namely, templates [1], ClassSheets [3, 6] and relational models [5] and also techniques to infer models from (legacy) spreadsheet data [1, 3].

Although all these works claim that a MDE approach improves end-users productivity, the reality is that there is no detailed evaluation study to support this idea. In this paper, we present an empirical study that we have conducted with the aim of analyzing the practical influence that models have on productivity. We consider two different model-based SS, as proposed in [4, 5]. We assess the productivity in introducing, updating and querying data in those two model-based SS and in a traditional one.

Our study is necessary and useful: it is based on a sound experimental setting which allows us to draw sound conclusions and directions for further studies on the same topic. With it, we wish to answer the following research questions:

RQ1 Do end users introduce fewer errors when they use one of the model-based spreadsheet versus the *original* unmodified spreadsheet?

RQ2 Are end users more efficient using the model-based ones?

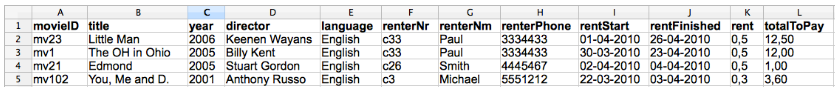
* Supported by Fundação para a Ciência e a Tecnologia, grants no. SFRH/BD/30231/2006, SFRH/BPD/46987/2008 and PTDC/EIA-CCO/108613/2008.

RQ3 Do particular models lead to fewer errors in particular tasks?

We used a within subjects design, where each participant received a task list for each of three spreadsheet environments. Participants were asked to do various tasks in each spreadsheet, for example: data entry, modifications to existing data, and calculations of the data in the spreadsheet. They were encouraged to work as quickly as possible, but were not given time limits for any specific spreadsheet.

2 Model-based Spreadsheets

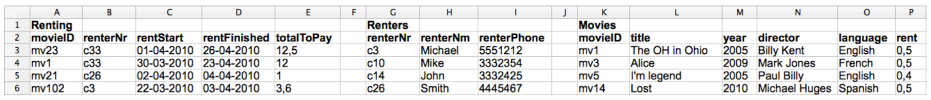
Two different techniques to tackle the problem of preventing errors in SS have been proposed [4, 5]. In order to introduce them we will rely on the spreadsheet shown in Fig. 1, which represents a movie renting system (labels should be self-explicative).



	A	B	C	D	E	F	G	H	I	J	K	L
1	**movieID**	**title**	**year**	**director**	**language**	**renterNr**	**renterNm**	**renterPhone**	**rentStart**	**rentFinished**	**rent**	**totalToPay**
2	mv23	Little Man	2006	Keenen Wayans	English	c33	Paul	3334433	01-04-2010	26-04-2010	0,5	12,50
3	mv1	The OH in Ohio	2005	Billy Kent	English	c33	Paul	3334433	30-03-2010	23-04-2010	0,5	12,00
4	mv21	Edmond	2005	Stuart Gordon	English	c26	Smith	4445467	02-04-2010	04-04-2010	0,5	1,00
5	mv102	You, Me and D.	2001	Anthony Russo	English	c3	Michael	5551212	22-03-2010	03-04-2010	0,3	3,60

Fig. 1: Part of a spreadsheet representing a movie renting system.

The Refactored Model: The spreadsheet in Fig. 1 shows an instance of a renting system containing redundant information: for example, client `Paul`'s information appears twice! This kind of redundancy makes maintenance complex and error-prone. A mistake is easily made (for example, by mistyping a name) and thus corrupting the data. The same information can be stored without redundancy: in the database realm, techniques for data normalization, based on the exploitation of functional dependencies (FDs) inherent in the data, are commonly used to minimize duplication of information and improve data integrity [2]. We have adapted these techniques to work with spreadsheets: from the spreadsheet data we infer a set of normalized FDs, and from them, we compute a relational model [5]. A spreadsheet respecting such model is shown in Fig. 2.



	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P
1	**Renting**						**Renters**				**Movies**					
2	**movieID**	**renterNr**	**rentStart**	**rentFinished**	**totalToPay**		**renterNr**	**renterNm**	**renterPhone**		**movieID**	**title**	**year**	**director**	**language**	**rent**
3	mv23	c33	01-04-2010	26-04-2010	12,5		c3	Michael	5551212		mv1	The OH in Ohio	2005	Billy Kent	English	0,5
4	mv1	c33	30-03-2010	23-04-2010	12		c10	Mike	3332354		mv3	Alice	2009	Mark Jones	French	0,5
5	mv21	c26	02-04-2010	04-04-2010	1		c14	John	3332425		mv5	I'm legend	2005	Paul Billy	English	0,4
6	mv102	c3	22-03-2010	03-04-2010	3,6		c26	Smith	4445467		mv14	Lost	2010	Michael Huges	Spanish	0,5

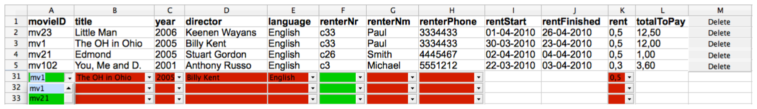
Fig. 2: Part of a refactored spreadsheet representing a movie renting system.

The obtained data organization solves two well-known database problems, namely *update anomalies* and *deletion anomalies* [2]. The former occurs when we change information in one tuple but leave the same information unchanged in others, e.g., if a user changes the rent per day of `mv23` from `0.5` to `0.6`. This value occurs only once in the modular spreadsheet, where it must be updated. The latter happens when we delete a tuple and lose other information as a side effect, e.g., if a user deletes the rent in row 3 in the original spreadsheet all the information concerning movie `mv1` is eliminated.

Since we know the data relations and relationships, we can generate SS that respect them, and thus, help end users. For example, in the *renter* table, the generated spreadsheet should not allow the user to introduce two renters with the same number.

The new spreadsheet improves modularity and detects the introduction of incorrect data, and also eliminates redundancy; this should help end users commit less errors.

The Visual Model: In [4], a technique to enhance a system with mechanisms to guide end users to introduce correct data was proposed. Using the relational database schema induced by the data we construct an environment that respects that schema. For example, for the movie spreadsheet, the system must not allow the introduction of two different movies with the same number. Instead, it offers to the user a list of possible movies, such that the value to fill in the cell can be chosen. This new spreadsheet, that we show in Fig. 3, also includes advanced features which provide information to the end user about correct data that can be introduced.



	A	B	C	D	E	F	G	H	I	J	K	L	M
1	movieID	title	year	director	language	renterNr	renterNm	renterPhone	rentStart	rentFinished	rent	totalToPay	Delete
2	mv23	Little Man	2006	Keenen Wayans	English	c33	Paul	3334433	01-04-2010	26-04-2010	0,5	12,50	Delete
3	mv1	The OH in Ohio	2005	Billy Kent	English	c33	Paul	3334433	30-03-2010	23-04-2010	0,5	12,00	Delete
4	mv21	Edmond	2005	Stuart Gordon	English	c26	Smith	4445467	02-04-2010	04-04-2010	0,5	1,00	Delete
5	mv102	You, Me and D.	2001	Anthony Russo	English	c3	Michael	5551212	22-03-2010	03-04-2010	0,3	3,60	Delete
31	mv1	The OH in Ohio	2005	Billy Kent	English						0,5		
32	mv1												
33	mv21												

Fig. 3: Part of a visual spreadsheet representing a movie renting system.

We consider three types of advanced features: (bidirectional) auto-completion of column values, non-editable columns and safe deletion of rows.

3 Analyzing End-users Performance

Effectiveness Each participant was handed 3 different lists of tasks to perform on 3 different spreadsheets, each of which constructed under a different model. We have graded their performance under each model, obtaining the results in Fig. 4. We notice that no model is neither the best nor the worst for all spreadsheets. Nevertheless, the results seem to indicate that models from [3] and [5] are not effective in reducing the number of errors, since one of them is always getting the lowest scores. This intuition deserves further development.

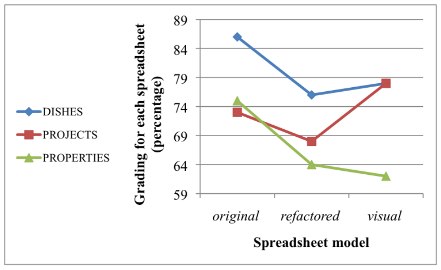


Fig. 4: Global effectiveness results.

One may argue that *original* is the *model* that users are more accustomed to. Nevertheless, we remark that the more complex models *refactored* and *visual* where given no explanation; a part of our study was also to learn whether they could live on their own.

Our next step was to investigate whether the (apparent) poor results obtained by complex models are due to their own nature or if they result from participants not having understood them. In order to realize this, we studied the participations that did not achieve a score of at least 50%: 0% in *original*, 25% in *refactored* and 21% in *visual*. While in *original* no participation was graded under 50%, this was not the case for *refactored* and *visual*; this may have degraded their overall average results. For these participations, we analyzed the questionnaire that participants were asked to fill in after the session. The classifications for the post session questionnaires, for participations in the study that were graded under 50% is 24% for *refactored* and 31% for *visual*.

These results show that participants obtaining poor gradings on their effectiveness, also got extremely poor gradings for their answers to the questions assessing how they understood (or not) the models. Indeed, we can see that they were not, in average, able to answer correctly to (at least) two thirds of the questions raised in the post session questionnaire. From such results we can read that (roughly) a quarter of participants was not able to understand the more complex models used in the study, which might have caused a degradation of the global effectiveness results for these models. This also suggests that if these models are to be used within an organization, it is necessary to take some time to introduce them to end users in order to achieve maximum effectiveness.

Effectiveness by Task Type: Next, we wanted to realize how effective models are to perform the different types of tasks that we proposed: *insertion*, *edition* and *statistics*.
i) Data insertion: the *original* model was the most effective, for all 3 spreadsheets, being closely followed by *refactored* and *visual* for DISHES, and by *visual* for PROJECTS. The *refactored* model, for PROJECTS, and *refactored* and *visual*, for PROPERTIES, proved not to be competitive for data insertion. Again, we believe that this in part due to the lack of introduction to these models: the insertion of new data is the task that most likely benefits from understanding them, and also the one that can be otherwise most affected. This is confirmed by the effectiveness results observed for other task types, that we present next.

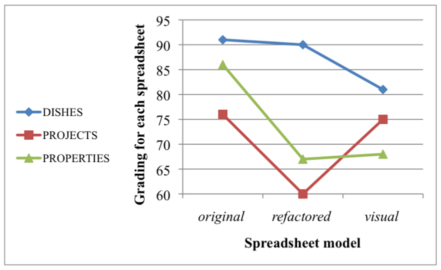


Fig. 5: Effectiveness results for insertion.

ii) Data edition once a spreadsheet is populated, we can effectively use models to edit it. This is the case of *refactored* for PROJECTS and for PROPERTIES. *original* is the most effective in editing for DISHES. *visual* is comparable to *refactored* for DISHES, but for other spreadsheets, it always achieves the lowest scores.
iii) Statistics: we can see that *visual* obtained the best results for DISHES, and that *refactored* obtained the best results for both spreadsheets PROJECTS and PROPERTIES. We can also see that all models obtained the worst results for exactly one spreadsheet.

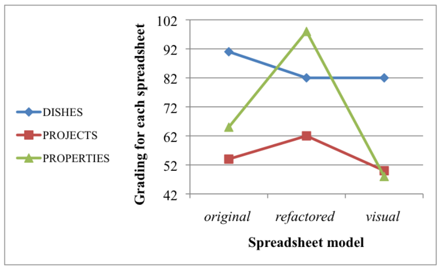


Fig. 6: Effectiveness results for edition.

Results from $i)$, $ii)$ and $iii)$ confirm that the models are competitive against the *original* model. On the other hand, these results allow us to draw some new conclusions: if the models are going to be used within an organization, it may not always be necessary to introduce them prior to their use. Indeed, if an organization mostly edits spreadsheet data or computes new values from such data, and does not insert new data, then the models, and specially *refac-*

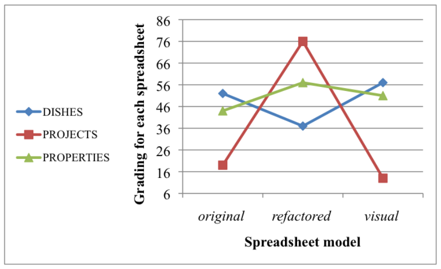


Fig. 7: Effectiveness results for statistics.

tored, may deliver good results even when they are not explicitly explained (as it was the case in our study). These results also show that it is in the data insertion tasks that the models need to be better understood by end users in order to increase effectiveness.

Efficiency We started by measuring, for each participant, and for each spreadsheet, the time elapsed from the moment participants started reading the list of tasks to undertake until the moment they completed the tasks proposed and moved on to a different spreadsheet or concluded the study. We are able of calculating these times by looking at the individual screen activity that was recorded during the study, for each participant: the participant stopping interacting with the computer signals the end of his/her work on a spreadsheet. The measured period therefore includes the time that participants took trying to understand the models they received each spreadsheet in. Fig. 8 presents the average of the overall times, for each spreadsheet and for each model.

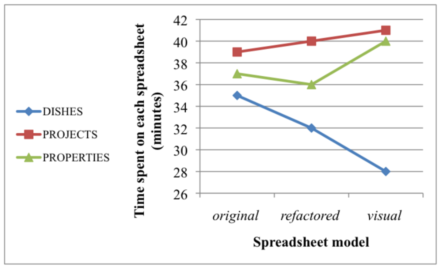


Fig. 8: Global efficiency results.

We can see that *refactored* and *visual* are competitive against *original*. Indeed, participants performed fastest for DISHES in *visual*, and fastest for PROPERTIES in *refactored*. The *original* model got the best result for PROJECTS. Again, note that no introduction to *refactored* or *visual* preceded the study. Therefore, it is reasonable to assume that, for these models, the results observed include some time overhead. In an attempt to measure this overhead we extracted some more information out of the results of our study: we measured the time elapsed from the moment participants started reading, for each spreadsheet, the list of tasks to perform, until the moment they actually began editing the spreadsheet. We assume that this period corresponds exactly to the overhead of understanding each model. The average results obtained are presented in Table 1.

There is a constant average overhead of 2 minutes for almost all models and all spreadsheets, with the most significant exceptions occurring for *refactored*, for both DISHES and PROJECTS. In these cases, we can clearly notice an important time gap, which provides some evidence that *refactored* is most likely the hardest model to understand. This also comes in line with previous indications that the merits of the models can be maximized if we take the time to explain them to end users. For the particular case of efficiency, this means that the results shown in Fig. 8 could be further improved for the more complex models, and particularly for *refactored*.

	original	*refactored*	*visual*
DISHES	$2'$	$6'$	$1'$
PROJECTS	$2'$	$4'$	$2'$
PROPERTIES	$2'$	$2'$	$2'$

Table 1: Average overhead results.

4 Conclusions

We have presented the results of an empirical study that we conducted in order to assess the practical interest of models for spreadsheets. From the preparation of the study, from running it and from its results, we can summarize our main contributions as follows: $i)$ we have shown that MDE techniques can be adapted for end-users software; $ii)$ we proved empirically that models can bring benefits for spreadsheet end users; $iii)$ we proposed a methodology that can be reused in studies similar to ours.

Now, we seek to answer our initial research questions.

RQ1 Our observations indicate that model-based spreadsheets can improve end-user effectiveness. Even if this is not always the case, our results also indicate that deeper insight on the spreadsheet models is required to maximize effectiveness.

RQ2 We observed that, frequently, the more elaborate spreadsheet models allowed users to perform faster. Nevertheless, we were not fully able of isolating the time that participants took trying to understand the models they were working with. So, we believe that the observed efficiency results could also be better for *refactored* and *visual* if they had been previously introduced.

RQ3 Although this was not observed for inserting tasks, for editing and querying data the models did help end users. Furthermore, the results seem to indicate that the inserting data task is the one that benefits the most from better understanding the models.

With this study we have shown that there is potential in MDE techniques for helping spreadsheet end users. The study of these techniques for professional users of spreadsheets seems a promising research topic. Moreover, the use of MDE techniques in other non-professional softwares should also be investigated.

References

1. Abraham, R., Erwig, M.: Inferring templates from spreadsheets. In: Proc. of the 28th Int. Conference on Software Engineering. pp. 182–191. ACM, New York, NY, USA (2006)
2. Codd, E.F.: A relational model of data for large shared data banks. Communications of the ACM 13, 377–387 (June 1970)
3. Cunha, J., Erwig, M., Saraiva, J.: Automatically inferring classsheet models from spreadsheets. In: Proc. of the 2010 IEEE Symposium on Visual Languages and Human-Centric Computing. pp. 93–100. IEEE Computer Society, Washington, DC, USA (2010)
4. Cunha, J., Saraiva, J., Visser, J.: Discovery-based edit assistance for spreadsheets. In: Proc. of the 2009 IEEE Symposium on Visual Languages and Human-Centric Computing. pp. 233–237. IEEE Computer Society, Washington, DC, USA (2009)
5. Cunha, J., Saraiva, J., Visser, J.: From spreadsheets to relational databases and back. In: PEPM '09: Proceedings of the 2009 ACM SIGPLAN Workshop on Partial Evaluation and Program Manipulation. pp. 179–188. ACM, New York, NY, USA (2009)
6. Engels, G., Erwig, M.: ClassSheets: Automatic generation of spreadsheet applications from object-oriented specifications. In: Proceedings of the 20th IEEE/ACM International Conference on Automated Software Engineering. pp. 124–133. ACM, New York, NY, USA (2005)
7. Nardi, B.A.: A Small Matter of Programming: Perspectives on End User Computing. MIT Press, Cambridge, MA, USA (1993)
8. Panko, R.R.: Spreadsheet errors: What we know. What we think we can do. Proceedings of the Spreadsheet Risk Symposium, European Spreadsheet Risks Interest Group (July 2000)
9. Rajalingham, K., Chadwick, D., Knight, B.: Classification of spreadsheet errors. European Spreadsheet Risks Interest Group (EuSpRIG) (2001)
10. Scaffidi, C., Shaw, M., Myers, B.: Estimating the numbers of end users and end user programmers. In: Proceedings of the 2005 IEEE Symposium on Visual Languages and Human-Centric Computing. pp. 207–214. IEEE Computer Society, Washington, DC, USA (2005)