An Empirical Study on the Influence of Different Spreadsheet Models on End-Users Performance

Jácome Cunha*
Universidade do Minho
Portugal
jacome@di.uminho.pt

Laura Beckwith
HCIResearcher
Denmark
beckwith@hciResearcher.com

João Paulo Fernandes†
Universidade do Minho
Portugal
jpaulo@di.uminho.pt

João Saraiva
Universidade do Minho
Portugal
jas@di.uminho.pt

August 13, 2010

Abstract

Spreadsheets are widely used by end users, and studies have shown that most end-user spreadsheets 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 systematic empirical study to assess the effectiveness and efficiency of this approach. A set of spreadsheet end users worked with two different model-based spreadsheets, and we present and analyze the results achieved.

1 Introduction

Spreadsheets can be viewed as programming environments for non-professional programmers, the so-called "end users", especially for developing business applications. End-user programmers vastly outnumber professional programmers creating every year hundreds of millions of spreadsheets [19]. As numerous studies have shown, this high rate of production is accompanied by an alarming high rate of errors [11, 14, 16]. Some studies report that up to 90% of real-world spreadsheets contain errors [17].

In order to overcome these limitations of spreadsheets, a considerable amount of research has been recently done by the human computer interaction community [1, 5, 6, 7, 9, 10]. One of the promising solution advocates the use of a Model-Driven Engineering (MDE) approach to spreadsheets. In such an approach, a business model

*Supported by Fundação para a Ciência e Tecnologia, grant no. SFRH/BD/30231/2006.
†Supported by Fundação para a Ciência e Tecnologia, grant no. SFRH/BPD/46987/2008.

of the spreadsheet data is defined, and then end users are guided to introduce data that conforms to the defined model [7]. Indeed, several models to represent the business logic of the spreadsheet have been proposed, namely, templates [1, 10], ClassSheets [5, 9], relational models [6]. Several techniques to infer such models from a (legacy) spreadsheet data have also been studied [5].

Although, all these works claim that using a MDE approach improves end users productivity, the reality is that there is no detailed evaluation/usability study that supports this idea. In this paper, we present in detail an empirical study that we have conducted with the aim of analyzing the practical influence of using models in end users spreadsheet productivity. In this study we consider two different model-based spreadsheets, as proposed in [6, 7]. We assess the productivity of end users when introducing and updating data in those two model-based spreadsheets and in a traditional one.

In this paper 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?

RQ3 Do particular models support particular tasks better, leading to fewer errors in those tasks?

The study we conduct to answer these questions is necessary and useful, since it is based on a sound experimental setting and thus allow us to draw sound conclusions for further studies how on to improve spreadsheet end users productivity.

This paper is structured as follows: first, we start by presenting the model-based spreadsheet we will consider throughout the paper. Next, e describe the design of our study. After that, we present and analyze in detail the results of our study, and we analyze several threats to validity. Finally, we give our conclusions.

2 The Models

There have been proposed two different techniques to tackle the problem of preventing errors in spreadsheets [6, 7]. Each of these techniques introduces a new model for the execution of spreadsheets, that we explain in detail in this section. In order to introduce this work, we will rely on the spreadsheet illustrated in Figure 1. This spreadsheet is used to simulate a simple movie renting system, that registers movies, renters and rents. Labels used in the spreadsheet should be self-explicative.

2.1 The Refactored Spreadsheet Model

Spreadsheets are usually created by a single end-user, without planning ahead of time for maintainability or scalability. Still, after their initial creation, many spreadsheets turn out to be used for storing and processing increasing amounts of data and supporting increasing numbers of users over long periods of time. In these cases concepts from

	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
6	mv23	Little Man	2006	Keenen Wayans	English	c26	Smith	4445467	02-12-2009	04-04-2010	0,5	61,50
7	mv23	Little Man	2006	Keenen Wayans	English	c14	John	3332425	12-04-2010	16-04-2010	0,5	2,00
8	mv3	Alice	2009	Mark Jones	French	c33	Paul	3334433	12-04-2010	23-04-2010	0,5	5,50
9	mv5	I'm legend	2005	Paul Billy	English	c33	Paul	3334433	05-04-2010	06-04-2010	0,4	0,40
10	mv102	You, Me and D.	2001	Anthony Russo	English	c26	Smith	4445467	22-03-2010	25-03-2010	0,3	0,90

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

databases can be applied, but how can we best do that for end users? How can we, as researchers and software developers, bring those best practices reliability techniques to end users?

In [6], the authors have developed techniques for transitions between spreadsheets and relational databases. The basis of these techniques is the fundamental insight that spreadsheets and relational databases are formally connected by a data refinement relation.

The spreadsheet illustrated in Figure 1 defines a valid model to represent the information of the renting system. However, it contains redundant information. For example, the information about the client `Paul` appears four times in the spreadsheet! This kind of redundancy makes the maintenance and update of the spreadsheet complex and error-prone, specially for end users. 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 fact, in the database community, techniques for database normalization are commonly used to minimize duplication of information and improve data integrity [8, 20]. Database normalization is based on the detection and exploitation of functional dependencies inherent in the data [3]. An illustration of part of the spreadsheet that can be obtained using these techniques is presented in Figure 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
7	mv23	c26	02-12-2009	04-04-2010	61,5		c33	Paul	3334433		mv19	The Last Song	2010	Julie Anne Robinson	French	0,4
8	mv23	c14	12-04-2010	16-04-2010	2						mv21	Edmond	2005	Stuart Gordon	English	0,5
9	mv3	c33	12-04-2010	23-04-2010	5,5						mv23	Little Man	2006	Keenen Wayans	English	0,5

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

The obtained modularity solves two well-known problems in databases, namely the *update anomalies* and the *deletion anomalies* [20]. The former problem occurs when we change information in one tuple but leave the same information unchanged in the others. In our example, this may happen if the user changes the rent per day of move number `mv23` from `0.5` to `0.6`. In the modular spreadsheet that value occurs only once in the movie table and so that problem will never occur. The latter problem happens when we delete some tuple and we lose other information as a side effect. For example, if the user deletes the rent in row 3 in the original spreadsheet all the information concerning movie `mv1` is eliminated.

Since we have a deep knowledge about the relations and relationships in the data,

we can generate spreadsheets that respect them, and thus, help end users. For example, in the *renter* table, the generated spreadsheet will not allow the user to introduce two renters with the same number, that is, the same *renterNr*. If that error occurs the spreadsheet's system should warn the user as shown in Figure 3. Obviously, it is not possible to perform this validation in the original spreadsheet.

G	H	I	
Renters			
renterNr	**renterNm**	**renterPhone**	
c3	Michael	5551212	
c10	Mike	3332354	ERROR
c14	John	3332425	
c26	Smith	4445467	
c33	Paul	3334433	
c10			ERROR

Figure 3: In the generated spreadsheet, if the user introduces a row in the renter table with a previously used code the spreadsheet will immediately produce an error.

The refactored spreadsheet not only improves modularity and detects the introduction of incorrect data, but also eliminates redundancy: indeed, the redundancy present in the original spreadsheet has been eliminated. As expected, the information about renters (and movies) occurs only once. This features should help the end users to commit less errors.

From now on, this model of spreadsheet will be referred as the *refactored* model.

2.2 The Visual Spreadsheet Model

Recent advances in programing languages extend naive editors to powerful language-based environments [12, 13, 18, 21]. Language-based environments use *knowledge* of the programming language to provide the users with more powerful mechanisms to develop their programs, helping them being more productive. This knowledge is based on the *structure* and the *meaning* of the language. Having this knowledge about a language, the language-based environment is not only able to highlight keywords and beautify programs, but it can also detect features of the programs being edited that, for example, violate the properties of the underlying language. Furthermore, a language-based environment may also give information to the user about properties of the program under consideration. Consequently, language-based environments guide the user in writing correct programs.

In [7], the authors proposed a technique to enhance a spreadsheet system with mechanisms to guide end users to introduce correct data. Using the relational database schema constructed as explained in the previous section we construct a spreadsheet environment that respects that schema. For example, for the movie spreadsheet it does not allow the user to introduce two different movies with the same number *movieID*, avoiding the committing of errors by end users. Instead, it offers to the user a list

of possible properties, such that he can choose the value to fill in the cell. This new spreadsheet, that we show in Figure 4, also includes advanced features which provide information to the end-user about correct data that can be introduced.

We consider three types of advanced features:

- Bidirectional auto-completion of column values: based on the relational schema, we know that some columns (consequents) depend on another column (antecedents). Both antecedent and consequent columns have *combo boxes* that allow users to choose values instead of writing them. Using this knowledge we created a mechanism that automatically fills in consequent column when antecedent columns are filled in. When values are written in consequent columns, the values in the antecedent columns are filtered, showing only the ones that can imply the chosen consequents. Using the bidirectional auto-completion feature the spreadsheet system guarantees that the end-user does not introduce data that violates the relational model inferred.

- Non-editable columns: this feature prevents the end user from editing columns that depend on antecedent columns since this could break the relationship. Note that, such columns are automatically filled in by selecting the corresponding antecedent.

- Safe deletion of rows: this feature warns if, by deleting a selected row some critical information is lost.

Like in modern programming language environment, the refactored spreadsheet system also offers the possibility of using traditional editing, that is, the introduction of data by editing each of the columns. When using traditional editing the end user is able to introduce data that may violate the relational database model inferred from the previous spreadsheet data. The spreadsheet environment includes a mechanism to re-calculate the relational database model after traditional editing. This new relational model is used to guide the end user in future non-standard editing of the spreadsheet.

	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**	
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
6	mv23	Little Man	2006	Keenen Wayans	English	c26	Smith	4445467	02-12-2009	04-04-2010	0,5	61,50	Delete
7	mv23	Little Man	2006	Keenen Wayans	English	c14	John	3332425	12-04-2010	16-04-2010	0,5	2,00	Delete
26	mv23	Little Man	2006	Keenen Wayans	English	c26	Smith	4445467	07-04-2010	09-04-2010	0,5	1,00	Delete
27	mv76	Clash of the Titans	2009	Louis Leterrier	English	c14	John	3332425	01-04-2010	15-04-2010	0,3	4,20	Delete
28	mv21	Edmond	2005	Stuart Gordon	English	c33	Paul	3334433	01-09-2009	16-04-2010	0,5	113,50	Delete
29	mv19	The Last Song	2010		French	c10	Mike	3332354	02-04-2010	21-04-2010	0,4	7,60	Delete
30	mv23	Little Man	2006	Keenen Wayans	English	c10	Mike	3332354	01-04-2010	13-04-2010	0,5	6,00	Delete
31	mv1	The OH in Ohio	200	Billy Kent	English						0,5		
32	mv1												
33	mv21												
34	mv102												
35	mv23												
36	mv23												

Figure 4: Part of a visual spreadsheet representing a movie renting system.

From now on, this model of spreadsheet will be referred as the *visual* model.

3 Study Design

As suggested in [15] we organized the study as follows:

1. *Formulating hypothesis to test*: our first step was to reason about the hypothesis. We spent a considerable amount of time organizing our ideas and finally formulating the hypothesis presented in this work (in the previous section).

2. *Observing a situation*: next, we gathered participants willing to be involved in the study and we ran the study itself, once we got enough and appropriately qualified participants. During the study, we screen casted the participants' computers and afterwards we collected the spreadsheets they worked on.

3. *Abstracting observations into data*: we then computed a series of statistics, that we present in detail in section **Analyzing end users performance**, over the spreadsheets participants developed during the study: we graded their performance and measured the time they took to perform the proposed tasks. All the data we computed is available at the *SSaaPP* web page[1]: `http://ssaapp.di.uminho.pt`.

4. *Analyzing the data*: the enormous collection of data that we gathered was later systematically analysed. This analysis is also presented in this paper, in section **Analyzing end users performance**.

5. *Drawing conclusions with respect to the tested hypothesis*: based on the results we obtained, we have finally drawn some conclusions. We were also able to suggest some future research paths based on our work, which are presented in the **Conclusions** section.

Our study aimed to answer if participants were able to perform their tasks with more accuracy and/or faster given the experimental environments. 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.

3.1 Methodology

Participants started the study by filling out a background questionnaire. The background questionnaire collected their area of study and previous experience with spreadsheets, other programming languages and English comfort (the study was held in Portugal). An introduction to the study was given orally in English, this was explicitly not a tutorial for the different environments because the goal was to see if even without any introduction to the various experimental spreadsheets the participants would

[1]The proposed lists of tasks and the initial spreadsheets that participants received are also available in that same page.

still be able to understand and complete the spreadsheet tasks. Although tutorials or training are sometimes available in the real world, environment designs which can be used without such training are preferable to those which need training. The participants were asked to work as quickly and accurately as possible. Since the order of the spreadsheets was randomized, they were told that the other sitting around them might appear to be moving faster, but that some tasks were shorter than others. After 2 hours participants were stopped if they were not already finished. Following the tasks they had a post session questionnaire which contained questions assessing their understanding of the different spreadsheet environments, (3 for *refactored* and 4 for *visual*), plus their own preference. We also asked how confident they were that they had correctly completed the tasks in the three task lists. Correct answers could only be given by participants having understood the running models. Grading the questionnaires was done as follows: A correct answer receives total points; an incorrect answer receives zero points and an answer that is not incorrect nor (totally) correct receives half the points. We recorded the users screens using screen capture technology. At the end of the study the users completed spreadsheets were saved and graded for later analysis.

3.2 Participants

Recruitment was conducted through a general email message to the university, asking for students with spreadsheet experience and comfort with English. Of the hundreds that responded[2], participants were selected based on spreadsheet experience, comfort with English, and majors outside of computer science and engineering. In total, 38 participants finished the study with data we were able to use (25 females, 11 males, and 2 who did not answer about their gender). Two participants did not try to solve one of the proposed tasks; for these participants, we included in the study only the tasks they undertook. A few participants' machines crashed and therefore they were eliminated from the study. The majority of participants were between 20-29 years of age, with the remaining under 20. All were students at the university. About 2/3 were working on their Baccalaureate degree, the remaining on their Masters. None were studying computer science or engineering and the most represented majors were medicine, economics, nursing and biology. A variety that is good for representing the end-user population of spreadsheet users. Table 1 resumes the participants' data.

What	How many
Participants	38
Female	25
Male	11
Not answer	2
In baccalaureate	25
In masters	13

Table 1: Summary of the participants' data.

[2]There was a compensation involved.

3.3 Tasks

The tasks were designed to highlight both the areas which the spreadsheet environments may be an advantage in, and areas where spreadsheets are known to be problematic. The tasks were 1) add new information into the spreadsheet, 2) edit existing data in the spreadsheets and 3) do some calculations using the data in the spreadsheets. Figure 5 illustrates an example of a sheet that participants received containing data for inserting.

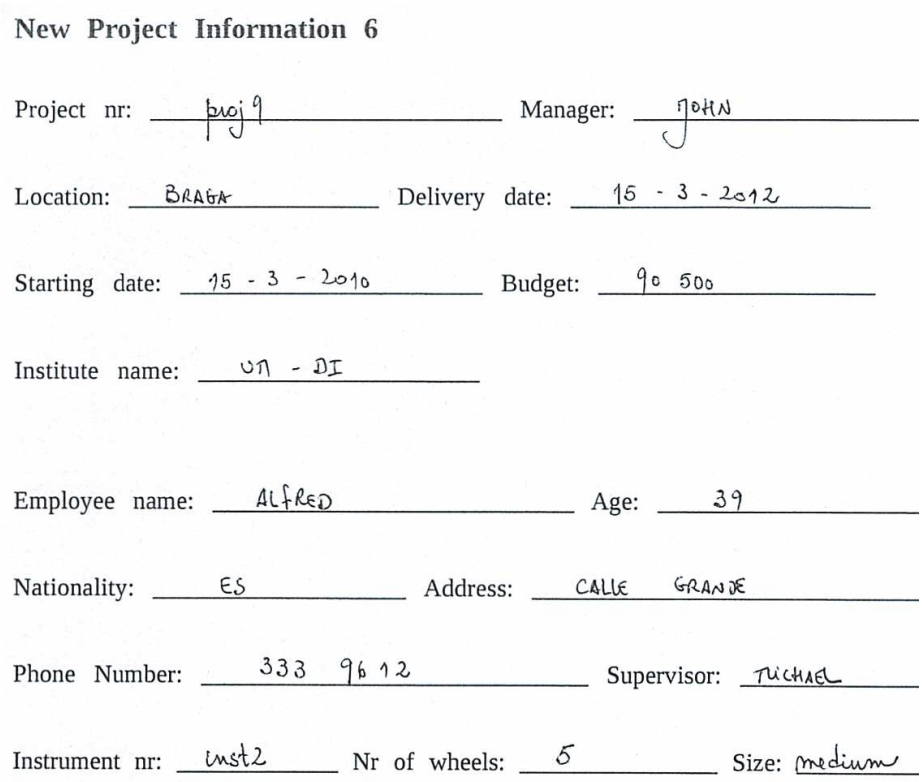

Figure 5: Example of a sheet that participants received containing data for inserting.

Some of the tasks asked the users to add many new rows of data, with the aim of a repetitive task being common in real-world situations. As we were designing the tasks, we imagined a type of data entry office scenario, where an office worker might receive on paper data which was initially filled out on a paper form and needed to be entered into a spreadsheet. This first task of data entry, in theory, should be fastest (and done with fewest entry errors) in the *refactored* spreadsheet environment. The second task, of making changes to existing data within a spreadsheet should also be easier within a *refactored* spreadsheet environment, since the change only needs to be made in one location, and therefore there would be less chance of forgetting to change it in one location. The final task was to do some calculations of the data in the spreadsheet,

such as averages, etc. This task was added because of the frequency of problems with formulas in spreadsheets.

In the task list for DISHES, 82% of the tasks consist of inserting new data, 3% are editing tasks and 15% involve calculations over the data in the spreadsheet. In the task list for PROJECTS, 98% of the tasks are for inserting new data, 1% for edition and 1% for calculations. Finally, for PROPERTIES, insertion data tasks are 67% of the total, whereas data edition and calculation tasks are 2% and 31% of the total, respectively.

Grading the participants' performance was done as follows. For tasks involving adding new data to the spreadsheet or performing calculations over spreadsheet data, whenever a participant executes a task as we asked him/her to, he/she is awarded 100% of the total score for that task; on the contrary, if the participant does not at all try to solve a particular task, he/she gets no credit for that. An intermediate situation occurs when participants try to solve a task, but fail to successfully conclude it in its entirety. In this case, the participant is awarded 50% of the score for that task. For tasks involving editing spreadsheet data, a value in the interval $0\% - 100\%$ is awarded according to the participants' success rate in such tasks (that we can accurately measure).

One of the spreadsheets used in the study, PROPERTIES, stores information about a house renting system and was adapted from [4]. This spreadsheet has information about renters, houses and their owners as well as the dates and prices of the rents.

A second spreadsheet, DISHES, contains information about sells of detergents to dish washers. Information about the detergents, prices and the stores where they are sold is present on this spreadsheets. This spreadsheet was adapted from [16].

The last spreadsheet, PROJECTS, stores information about projects, like the manager and delivery date, employees assigned to them and the instruments used. This spreadsheet was adapted from [2].

Table 2 presents the number of participants that worked on each spreadsheet and each model.

	original	*refactored*	*visual*	Total
DISHES	12	13	12	37
PROJECTS	11	13	13	37
PROPERTIES	14	11	13	38
Total	37	37	38	

Table 2: Number of participants that worked on each spreadsheet/model.

4 Analyzing End Users Performance

We divide the presentation of our empiric results under two main axes: Effectiveness and efficiency. In studying effectiveness we want to compare the three running models for the percentage of correct tasks that participants produced in each one. In studying efficiency we wish to compare the time that participants took to execute their assigned tasks in each of the different models.

4.1 Effectiveness

The results presented in this section report to the effectiveness of the models we consider.

Each participant was handed three different lists of tasks to perform on three different spreadsheets (DISHES, PROJECTS and PROPERTIES). Each of the spreadsheets, for the same participant, was constructed under a different model (*original*, *refactored* or *visual*).

For each spreadsheet, and for each model, we started by analyzing the average of the scores obtained by participants. The results presented in Figure 6 are the outcome of that analysis.

	original	*refactored*	*visual*
DISHES	86%	76%	78%
PROJECTS	73%	68%	78%
PROPERTIES	75%	64%	62%

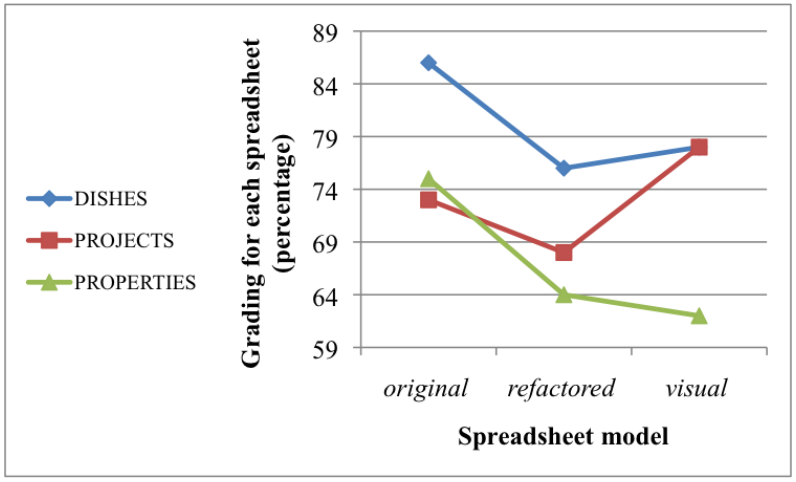

Figure 6: Global effectiveness results.

We notice that no spreadsheet model is the best for all spreadsheets, in the conditions tested by the study, and in terms of effectiveness. Indeed, we may even notice that spreadsheets in the traditional style, the *original* model, turned out to be the best for both the DISHES and PROPERTIES spreadsheets'. The *visual* model suited the best for the PROJECTS spreadsheet.

In the same line of reasoning, there is no worst model: *refactored* spreadsheets achieved the worst results for the DISHES and PROJECTS spreadsheets and the *visual* model got the lowest average scores for PROPERTIES. Neverthless, these results seem to indicate that the models that we have developed in [5] and [6] are not effective in reducing the number of errors in spreadsheets, since one of them is always the model

getting the lowest average scores. This first intuition, however, deserves further development.

For once, on the theoretical side, one may argue that the *original* spreadsheet model is, without a doubt, the *model* that end users are accustomed to. Recall that in the study, we opted to leave out participants with computer science backgrounds, who could be more sensible to the more complex models *refactored* and *visual*, prefering to investigate such models on traditional users of spreadsheets. On the other hand, we remark that this more complex models where given no introductory explanation; a part of our study was also to learn whether or not they could live on their own.

Our next step was then 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 fully realize this, we studied the participations that did not achieve a score of at least 50%, which are distributed by spreadsheet model as follows:

original	0%
refactored	25%
visual	21%

Figure 7: Participations graded under 50%.

While in the *original* model no participation was graded under 50%, we can see that this was not the case for *refactored* and *visual*, which may have degraded their overall average results. For these participations, we then analyzed the questionnaire that participants were asked to fill in after the session. In Figure 8, we present the average classifications, in percentage, for the post-session questionnaires, for participations in the study that were graded under 50%.

refactored	24%
visual	31%

Figure 8: Grading of post-session questionnaires for participations graded under 50%.

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 they had worked with. Indeed, we can see that such participants 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. Nevertheless, even having this introduction never occurred in our study, the results presented in this section show that the models we have developed are competitive in terms of effectiveness: At most they are 13% worst

than the *original* model, and for one of the spreadsheets studied, the *visual* model even got the best global effectiveness results.

4.1.1 Effectiveness by Task Type

Next, we wanted to realize how effective the models are for performing each of the different types of tasks that we have proposed to participants: Adding new information to a spreadsheet, or *data insertion*, changing the information on a spreadsheet, or *data edition* and performing some calculations, or *statistics*, over the spreadsheet data.

i) *Data insertion*

The results presented in Figure 9 show, for each model, how effective participants in the study were in adding new information to the spreadsheets that they were given.

	original	*refactored*	*visual*
DISHES	91%	90%	81%
PROJECTS	76%	60%	75%
PROPERTIES	86%	67%	68%

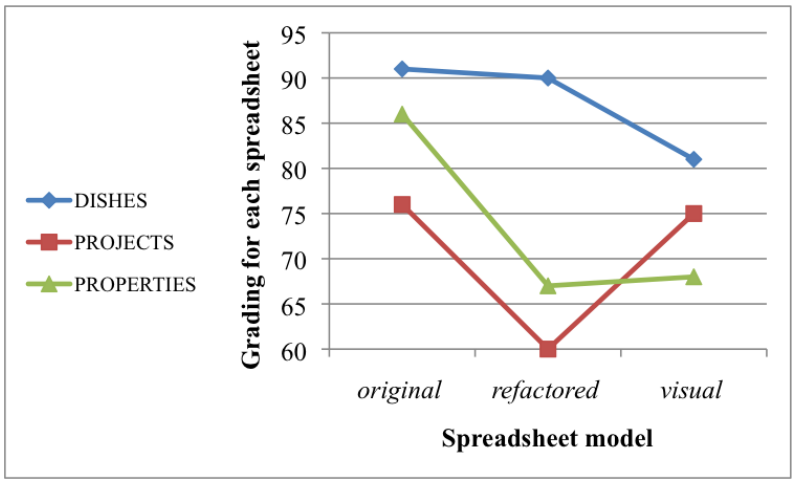

Figure 9: Effectiveness results for data insertion.

The *original* model revealed to be the most effective, for all three spreadsheets, being closely followed by *refactored* and *visual* for DISHES, and by *visual* for PROJECTS. The *refactored* model, for PROJECTS, and the models *refactored* and *visual*, for PROPERTIES, proved not to be competitive for data insertion, in the context of the study. Again, we believe that this in part due to these models not having been introduced previously to the study: The insertion of new data is the task that is most likely to benefit from totally understanding of the running

model, 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.

ii) *Data edition*

Now, we analyze the effectiveness of the models for editing spreadsheet data. The results presented in Figure 10 show that once a spreadsheet is populated, we can effectively use the models to edit its data. This is the case of the *refactored* model for PROJECTS and specially for PROPERTIES. The *original* model is the most effective in data editing for DISHES. The *visual* model is comparable to *refactored* for DISHES, but for all other spreadsheets, it always achieves the lowest scores among the three models.

	original	*refactored*	*visual*
DISHES	91%	82%	82%
PROJECTS	54%	62%	50%
PROPERTIES	65%	98%	48%

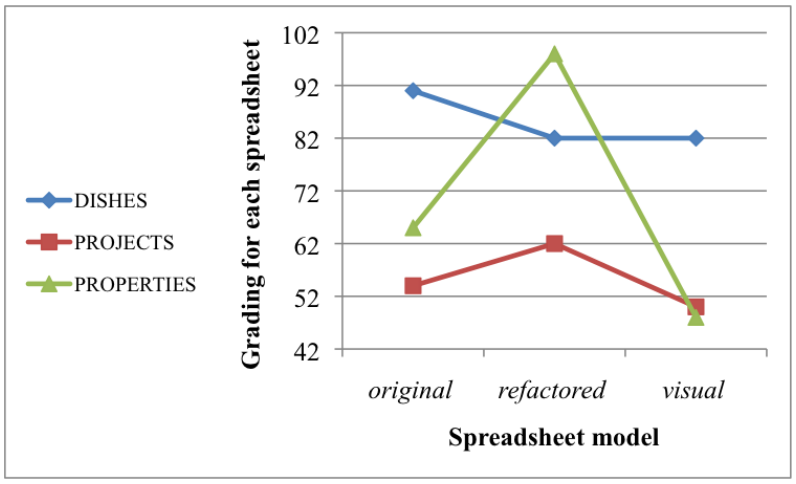

Figure 10: Effectiveness results for data edition.

iii) *Statistics*

Finally, we have measured the effectiveness of the models for performing calculations over spreadsheet data, obtaining the results shown in Figure 11.

We 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.

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

	original	*refactored*	*visual*
DISHES	52%	37%	57%
PROJECTS	19%	76%	13%
PROPERTIES	44%	57%	51%

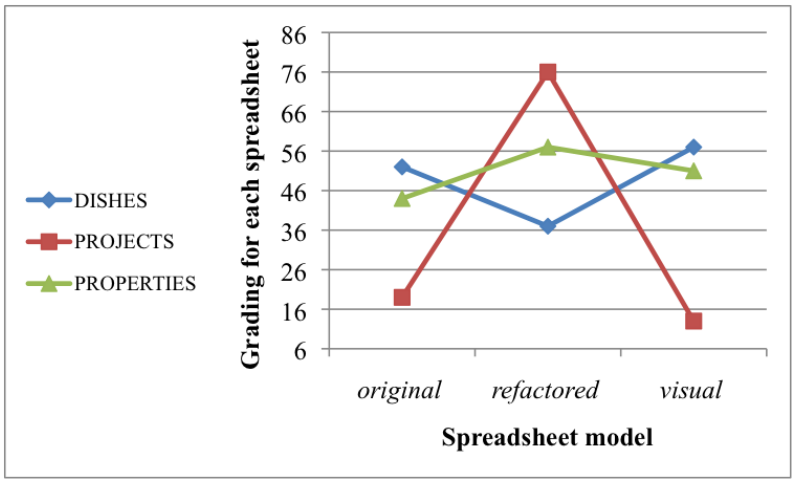

Figure 11: Effectiveness results for statistical calculations.

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 *refactored*, 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.

4.2 Efficiency

In this section, we analyze the efficiency results obtained, in our study, by the models that we have been considering in this paper.

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 for that particular spreadsheet 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 in trying to understand the models they received each spreadsheet in. Figure 12 presents the average of the overall times, for each spreadsheet and for each model.

	original	*refactored*	*visual*
DISHES	$35'$	$32'$	$28'$
PROJECTS	$39'$	$40'$	$41'$
PROPERTIES	$37'$	$36'$	$40'$

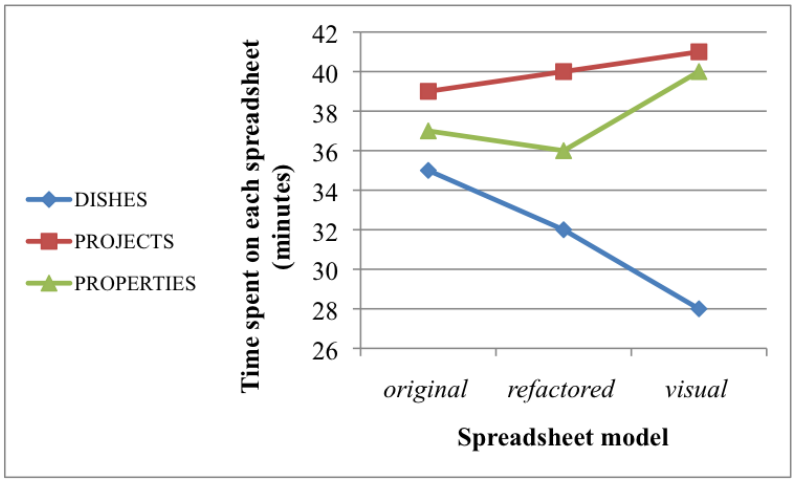

Figure 12: Global efficiency results.

We can see that the models *refactored* and *visual* are competitive in terms of efficiency against the *original* model. Indeed, participants performed fastest for the DISHES spreadsheets in the *visual* model, and fastest, by a marginal factor, for the PROPERTIES spreadsheet in the *refactored* model. The *original* model got the best efficiency measurements for the PROJECTS spreadsheets, also by a marginal factor. Again, we should stress out that no introduction to *refactored* or *visual* preceded the study. Therefore, it is reasonable to assume that, for these models, the results in Figure 12 include some time overhead. In an attempt to measure this overhead, which is a consequence of participants having to analyze a new model, we extracted some more information out of the results of our study (particularly from the participants' individual screen activity record). Indeed, 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 (obviously increased by the time spent reading the list of tasks, which we are not able of isolating further, but that should be constant for any spreadsheet model, since the task list does not change with the model). The average results obtained are presented in Figure 13.

We notice that there is a constant average overhead of 2 minutes for almost all models and all spreadsheets, with the most significant exceptions occurring for the *refactored* model, for both the DISHES and the PROJECTS spreadsheets. In these cases, we can clearly notice an important time gap, which provides some evidence that *refactored*

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

Figure 13: Average overhead results.

is most likely the hardest model to understand. This also comes in line with previous indications that the merits of the spreadsheet 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 Figure 12 could be further improved for the more complex models, and particularly for the *refactored* model.

5 Threats to Validity

As suggested by Perry *et al.* [15], we discuss three types of influences that might limit the validity of our study.

5.1 Construct Validity:

Do the variables and hypotheses of our study accurately model the research questions?

- *Measuring the time overhead*: when studying efficiency, we measured the overhead of understanding each model as the period of time that participants stopped interacting with a particular spreadsheet and started editing the following one. In this period, it might have been the case that participants, instead of being really focused on understanding the new model, took the time to do something else, like resting, for example. This could affect the conclusions that we draw, in terms of efficiency. However, during the entire study, participants where always supervised by at least two authors, who observed that this was not the case. Even if we were not able to spot a small number of such occurrences, the differences in the results computed should be minimal and therefore they should not affect our conclusions.

- *Original model*: In our study, we have used three spreadsheets that we have assumed to be in the *original* model. What we are saying is that these three spreadsheets are representative of the spreadsheets normally defined by end users. Although this set of spreadsheets may be too large to be represented by (any) three spreadsheets, we have taken DISHES,PROJECTS and PROPERTIES directly, or with small changes, from other works on general purpose spreadsheets [2, 4, 16].

5.2 Internal Validity:

Can changes in the dependent variables be safely attributed to changes in the independent variable?

- *Accuracy of the analysis*: Some of the inferences we make in this paper deserve further analysis. To some extent, we assume that our models could achieve better results if a tutorial has been given to the participants. In fact, we have no proof of this, but the evidences from the study seem to strongly indicate this fact. A new study is required to prove this, though.

- *Accuracy of measurements*: Each task proposed to participants was graded individually according to the participants' performance. For most of the cases, this was done automatically using OpenOffice Basic scripts. These scripts and their results were exhaustively tested and checked. The cases for which an automatic grading was not possible were carefully graded by hand. All grades were validated by two authors and were randomly re-checked. Since we have more than 1400 grades, it is virtually impossible to guarantee full grading accuracy. This could affect the results observed for dependent variables (efficiency and effectiveness) without really the independent variables (the models considered) having changed. Nevertheless, if imprecisions exist in the grades, they should be equally distributed by the three models and thus they should not affect the overall results or our conclusions.

  The measurement of times that lead to the results presented earlier was achieved by individually visualizing the screen cast made during the study for each participant. Being a manual task, and a repetitive one, this is subject to imprecisions. Also, not being able to visualize the actual participants' behavior now may lead us to imprecise measurements. We are confident that, even if the observed results are in fact subject to imprecisions, such imprecisions should be distributed evenly by all measurements and thus do not influence the efficiency results or the conclusions that we draw based on them.

5.3 External Validity:

Can the study results be generalized to settings outside the study?

- *Generalization*: In this study we used three different spreadsheets from different domains. We believe that the results can be generalized to other spreadsheets, although probably not to all. The models we developed are not restricted to any particular spreadsheet, and thus, the results should be the same if the the study was run with a different set of spreadsheets.

- *Industrial usage*: Participants in our study were students who were asked to simulate industrial activity: they received some data on paper that they had to register in a spreadsheet, and to further manipulate. Although we have tried to create conditions similar as possible to reality, it is likely that people could act differently in an industrial/real environment. Nevertheless, we believe that no spreadsheet or model should be affected in particular by this. Indeed, if this would really be the case, then it probably would affect all spreadsheets and all models in the same way and thus the overall results apply. We believe that if the study was conducted on an industrial environment, the conclusions should be similar.

6 Conclusions

In this paper, we have presented the results of an empirical study that we have conducted in order to assess the practical interest of models for spreadsheets.

According to [15], three topics deserve further analysis, namely, *accuracy of interpretation*, *relevance* of our study and its *impact*.

- Accuracy of interpretation: This study was prepared carefully and a significantly large number of end users participated in it. Our goal here was to guarantee that the results are not unknowingly influenced. For this it also contributes the fact that we make all the elements of this study available, both in this paper and online.
- Relevance: Model-Driven Engineering is one of the most significant research areas in software engineering. We adapted some techniques from this field to spreadsheets and showed that they can bring benefit not only for professional users but also for end users.
- Impact: Our first results show that MDE can bring benefits for spreadsheet end users. This is a promising research direction, that we believe can be further explored, particularly in contexts in all similar to the one of this paper.

From the preparation of the study, from running it and from its results, we can summarize our main contributions as follows:

- We have shown that MDE techniques can be adapted for end users software;
- We proved empirically that models can bring benefits for spreadsheet end users;
- We proposed a methodology that can be reused in studies similar to the one we have conducted;

Finally, we seek to answer the research questions that we presented in the introduction of this paper, which correspond exactly to the questions our study was designed to answer.

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

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. Indeed, we believe that the effectiveness results for *refactored* and *visual* could have been significantly better if these models had been preliminary presented to the participants of our study.

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

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 in trying to understand the models they were working with. So, we believe that the observed efficiency results could also have been better for *refactored* and *visual* if they had been previously introduced.

RQ3 Do particular models support particular tasks better, leading to fewer errors in those tasks?

Although this was not observed for editing tasks, the fact is that, 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.

Acknowledgments

We would like to thank all participants in our study for their enthusiasm and commitment. We would also like to thank Joost Visser for providing useful comments on drafts of this paper. This work was developed in the context of the research project *SpreadSheets as a Programming Paradigm*, `http://ssaapp.di.uminho.pt`, under FCT contract `PTDC/EIA-CCO/108613/2008`.

References

[1] R. Abraham and M. Erwig. Inferring templates from spreadsheets. In *ICSE '06: Proceedings of the 28th international conference on Software engineering*, pages 182–191, New York, NY, USA, 2006. ACM.

[2] R. Alhajj. Extracting the extended entity-relationship model from a legacy relational database. *Information Systems*, 28(6):597 – 618, 2003.

[3] C. Beeri, R. Fagin, and J. Howard. A complete axiomatization for functional and multivalued dependencies in database relations. In *Proc. of the ACM SIGMOD Int. Conf. on Management of Data*, pages 47–61, 1977.

[4] T. Connolly and C. Begg. *Database Systems, A Practical Approach to Design, Implementation, and Management*. Addison-Wesley, 3 edition, 2002.

[5] J. Cunha, M. Erwig, and J. Saraiva. Automatically inferring classsheet models from spreadsheets. In *VLHCC '10: Proceedings of the 2010 IEEE Symposium on Visual Languages and Human-Centric Computing (VL/HCC)*. IEEE Computer Society, 2010. to appear.

[6] J. Cunha, a. Saraiva, Jo and J. Visser. From spreadsheets to relational databases and back. In *PEPM '09: Proceedings of the 2009 ACM SIGPLAN Workshop on*

Partial Evaluation and Program Manipulation, pages 179–188, New York, NY, USA, 2009. ACM.

[7] J. Cunha, J. Saraiva, and J. Visser. Discovery-based edit assistance for spreadsheets. In *2009 IEEE Symposium on Visual Languages and Human-Centric Computing*, pages 233–237. IEEE, 2009.

[8] C. J. Date. *An Introduction to Database Systems*. Addison-Wesley, 1995.

[9] G. Engels and M. Erwig. ClassSheets: automatic generation of spreadsheet applications from object-oriented specifications. In D. Redmiles, T. Ellman, and A. Zisman, editors, *20th IEEE/ACM Int. Conf. on Automated Sof. Eng., Long Beach, USA*, pages 124–133. ACM, 2005.

[10] M. Erwig, R. Abraham, I. Cooperstein, and S. Kollmansberger. Automatic generation and maintenance of correct spreadsheets. In *ICSE '05: Proceedings of the 27th international conference on Software engineering*, pages 136–145, New York, NY, USA, 2005. ACM.

[11] EuSpRIG. European Spreadsheet Risks Interest Group, `http://www.eusprig.org/`, 2010.

[12] S. Holzner. *Eclipse*. O'Reilly, May 2004.

[13] M. Kuiper and J. Saraiva. Lrc - A Generator for Incremental Language-Oriented Tools. In K. Koskimies, editor, *7th International Conference on Compiler Construction*, volume 1383 of *LNCS*, pages 298–301. Springer-Verlag, April 1998.

[14] R. Panko. Spreadsheet errors: What we know. what we think we can do. *Proceedings of the Spreadsheet Risk Symposium, European Spreadsheet Risks Interest Group (EuSpRIG)*, July 2000.

[15] D. E. Perry, A. A. Porter, and L. G. Votta. Empirical studies of software engineering: a roadmap. In *ICSE '00: Proceedings of the Conference on The Future of Software Engineering*, pages 345–355, New York, NY, USA, 2000. ACM.

[16] S. G. Powell and K. R. Baker. *The Art of Modeling with Spreadsheets*. John Wiley & Sons, Inc., New York, NY, USA, 2003.

[17] K. Rajalingham, D. Chadwick, and B. Knight. Classification of spreadsheet errors. *European Spreadsheet Risks Interest Group (EuSpRIG)*, 2001.

[18] T. Reps and T. Teitelbaum. The synthesizer generator. *SIGSOFT Softw. Eng. Notes*, 9(3):42–48, 1984.

[19] C. Scaffidi, M. Shaw, and B. Myers. Estimating the numbers of end users and end user programmers. *VLHCC '05: Proceedings of the 2005 IEEE Symposium on Visual Languages and Human-Centric Computing*, pages 207–214, 2005.

[20] J. D. Ullman and J. Widom. *A First Course in Database Systems*. Prentice Hall, 1997.

[21] M. van den Brand, P. Klint, and P. Olivier. Compilation and Memory Management for ASF+SDF. In Stefan Jähnichen, editor, *8th International Conference on Compiler Construction*, volume 1575 of *LNCS*, pages 198–213, Mar. 1999.