INFORMATION MICROSYSTEMS

Jordi Pradel, Jose Raya and Xavier Franch
Agilogy, Universitat Politècnica de Catalunya (UPC), Barcelona, Spain

Keywords: Information Systems, Data Conceptual Models, Architectural Styles.

Abstract: Given their need to manage the information they have under control, organizations usually choose among two types of widely used IT solutions: 1) information systems based on databases (DBIS) that are powerful but expensive to develop and not much flexible; 2) spreadsheets, which threaten the integrity of the data and are limited in exploiting it. In this paper we propose a new type of IT solution, namely *information microsystems* (MicroIS), that aims at reconciling the best of these two worlds: the low development and maintenance costs, ease of use and flexibility of spreadsheets, with the structure, semantics and integrity of DBIS. The goal is not to replace any of the above two paradigms but to lie somewhere in between depending on the changing needs of the organization. From the various possible points of interest of this IT solution, the article will focus specifically on issues related to data management, introducing the conceptual model of MicroIS, the transformations and validations that can be done around it, and they way in which the structure of the information is inferred from the data that users provide.

1 INTRODUCTION

Nowadays, organizations of any kind require storing and processing a big amount of information of all types. With this goal, they may follow different options. On the one hand, the development of information systems (IS) allows managing the information in a highly efficient and secure way with purposes such as transaction processing, decision-making support, knowledge management, etc. Such systems usually are deployed as a collection of services (bespoke, acquired or licensed) over an architectural solution that relies on a (typically relational) data base. We label this type of solution with the acronym DBIS (Data Base-based Information System).

Another type of tool that has facilitated information processing in organizations is that of spreadsheets. Through an extremely simple and easy to understand paradigm and offering an almost unlimited flexibility concerning the contents of each table's cell, it becomes straightforward to create, at a minimum cost, small systems that allow storing information and solving, at least partially, the automation needs of the organization without being necessary to face a complex IT solution development process.

Both paradigms lie in opposite edges with respect to flexibility: whilst DBIS restrict the system entries that are valid, spreadsheets allow writing almost any value at any cell. This divergence results in a big difference concerning data integrity, that is almost complete in DBIS and virtually inexistent in spreadsheets. They are very different with respect to development cost: spreadsheets allow reusing the same generic user interface in all the applications, whilst in DBIS it is necessary either to develop the whole, or part of a former one, system, or to parameterise a generic system, being required in both cases to have some knowledge on software development and integration, which results in a high initial cost to be invested before obtaining the implied benefits. Last, another aspect equally important is the difference with respect to mobility: whilst spreadsheets may be ported without much effort from one platform to another, DBIS require a migration process that is not so straightforward.

The purpose of this paper consists on presenting a solution that allows building IS with the same properties than spreadsheets: easy of use, a minimal initial cost and short-term return on investment but that, unlike spreadsheets, supports evolution into an DBIS, with integrity rules, relationships among data, etc. The proposal will reflect the decisions made by two of the three authors in their daily work in a small IT solution-provider company (4 IT workers) that is specialized in deploying custom-made IT solutions to small enterprises that usually cannot afford the high cost of DBIS. In fact, the case study that will be used throughout the paper for illustration purposes reflects one of these real projects. Since our purpose is to facilitate the deployment of small IS, we have named this new type of IT solution as Information Microsystems (MicroIS for short). Fig. 1 summarises the main purpose of MicroIS as a compromise among DBIS and spreadsheets.

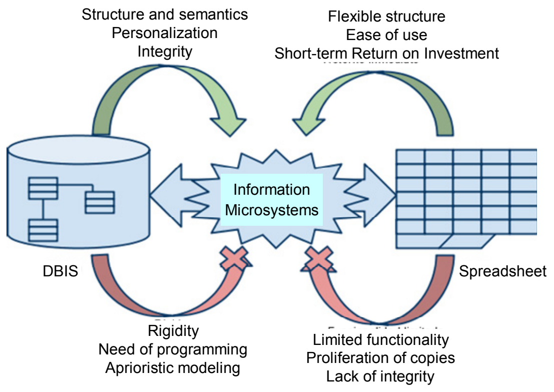

Figure 1: MicroIS: a trade-off among DBIS and spreadsheets.

As shown in this introduction, the problem faced by MicroIS is complex and for reasons of space we focus on this work in areas directly related to the issue of data structuring. In particular we address the two following features:

- Semi-automatic data modeling: The system shall, by appropriate heuristics, suggest the user possible improvements in the data schema from the analysis of the stored information so far, in a way that that a user without knowledge of conceptual modeling may add structure to his/her IS. Therefore, the data schema can evolve safely and easily, without affecting the data already stored.
- Non-intrusive integrity constraints: The system will know the data integrity constraints and will show and ask the user for those values that do not comply them, and eventually will provide suggestions for corrections, but these possible violations of the rules will never prevent the user to enter the data that they want. At every moment, the user will be able to locate and correct the inconsistencies if desired.

2 RELATED WORK

The dichotomy DBIS – Spreadsheets is well-known and has been addressed by several authors. The Dabble blog (2005) presents clearly the problematic of those DBIS whose utility does not guarantee the return on investment that its development requires and relates this fact with the concept of "long-tail" (Kraus, 2005), stating that there is a great deal of small IS that have not been developed as such for this reason. The conclusion, however, is very clear stating that the solution to this problem is not spreadsheet adoption.

Why, then, are spreadsheets used to implement this type of system? Baker, Foster-Johnson, Lawson, and Powell (2006) did an analysis of spreadsheets over a population of approximately 1600 individuals, noting that only 23% of respondents did some form of modeling prior to the development of the spreadsheet. It is reported that the absence of a need for aprioristic modeling facilitates the IS creation, because the user can start to enter data from the very beginning, and add formulas and formats as need, whilst they are discovered. Problems arise once information needs evolve, and inconsistencies start to appear or simply its daily use becomes cumbersome given the little support provided by the system for the introduction and exploitation of data (unlike DBIS, which normally have a presentation layer designed for that purpose). At this point, the spreadsheet solution cannot evolve further and the organization faces again the need of affording the development cost of an IS. The DabbleDB application aims at facilitating the handling of data and schema evolution. Despite sharing the motivation, we believe that DabbleDB is not the ideal choice for this type of systems because it requires an active connection to an Internet server and, therefore, is not usable in a great deal of situations. Moreover, it lacks the ability of self-organizing, so that the whole modelling responsibility falls on the user without assistance from the system.

An alternative solution would be the relational treatment of the information stored in a spreadsheet. Thus, Cunha, Saraiva and Visser (2009) proposed the HaExcel tool, which proposes a bidirectional correspondence between spreadsheets and relational databases (RDB). However, the objectives mentioned in the introduction of this paper, namely implicit modeling and non-intrusive integrity constraints, are not part of the HaExcel objectives. In Microsoft ADO.net the possibility of establishing this correspondence is also mentioned but with the toll of limiting significatively limiting the starting structure of the spreadsheet.

Another approach to the problem consists on reducing the cost of developing the IS through the automatic generation of applications, based on:

models, such as AndroMDA; code, as in OpenXava; or in existing databases such as Ruby On Rails. Despite their reported benefits, we believe that all these tools share a number of disadvantages derived from the use of RDB: the need for modelling from the start, that prevents users without development skills to implement their own solutions (thus their fail on overcoming "pragmatic" solutions as spreadsheets); lack of flexibility in the implementation of the scheme; and the difficulty of evolution, since the process of generating the application is unidirectional.

3 CASE STUDY

For illustration purposes, we present a simplified version of a real project developed by Agilogy, an application that manages the information of anthropometric data and the visits of customers to a small dietetics organization, Lalinia. Given this small size, Lalinia couldn't afford the development and exploitation of an DBIS due to the high cost of this type of solution. But on the other hand, Lalinia wouldn't accept the spreadsheet option due to the risks related to the low level of data integrity. The most fundamental factors for choosing MicroIS were:

- Reduced information. The volume of data to manage is limited enough as not to require a data base with rich functionality.
- Low development cost. Given the skeleton of the MicroIS solution type, creating a new MicroIS as the one for Lalinia becomes basically a customization/adaptation process.
- Flexibility. Since Lalinia didn't have a similar system before, the main stakeholders (basically, the dietist and the secretary) were not sure of knowing the appropriate system requirements in advance. Even more, the stability of data requirements along time was not guaranteed. As a result, it was considered mandatory to be able to add new structure to the information once new requirements are gradually discovered.
- Easy of use. It was a requirement that the application didn't interfere with the natural flow of a client's visit, so Lalinia accepted to allow incorrect data entry during the visit that would be correct at some later moment, or even could eventually remain incorrect.
- Data entry support. The system should be aware of the data semantics so as to make possible their input (choosing the values of an association from a combo box, help fulfilling the constraints bound to an attribute, etc.).

In Fig. 2, we show a possible scenario that illustrates the factors above mentioned. The two main types of information to store are represented in a tabular form: the personal information and the anthropometric data; in the second sheet each visit is represented using a row (item). It may be observed some of the situations that are typical in these MicroIS related to data structuring:

- Attributes which are specific to an item. For instance, there is an item (Luisa Fernández) that has a datum (the vacation home address) that, for the rest of clients, is rarely known. These data are not known in advance, i.e. each client may provide some particular data and Lalinia wants to be able to classify the information that is considered useful instead of simply maintaining it in a generic comment field. Also, it may happen that at some future moment, one of this datum becomes standard, either because a lot of clients provide it, or because Lalinia discovers some unexpected utility that increases its criticality.
- Document that has some potential errors. The client Juan Pérez has not given his telephone number but nevertheless Lalinia wants to keep the rest of data of this item whilst the missing information does not arrive.
- Lack of structure. Whilst the weight is a data that is sampled in each single visit, the height may be just measured the first time, except for the case of children and teenagers. But nevertheless, if the dietist decides to measure the height beyond the first visit, it must be possible to record the new measure.
- High variability. We may observe the different forms in which the clients provide their contact telephone number. Any attempt to anticipate to all the possible cases and determine a structure in advance is condemned to failure.

The proposed solution allows these situations to exist but at the same time provokes some vulnerability bound to the possible existence of inconsistencies. These inconsistencies may appear at the intra-sheet level (for instance, in the Anthropometric Data sheet we may observe a typographic mistake in the second client row, first name, that should be the same than the row above) or at the inter-sheet level (e.g., in this very sheet the first and last names of the client are copied and pasted manually from the contents written in the Personal Data sheet, therefore if some data is modified in the latter, the required modification in the former should be made manually, it is not enforced). These undesired situations must be also considered in the given solution.

Personal Data						
First name	Last name	Telephone	e-mail	Fax	Address	Vacation address
Jordi	Garcia Gil	934130000 ext. 512		+54875	Spring, 1	
Montse	Tolrà	934444444 10-14h 666666666 (mobile)	mtolra@ gmal.com		River, 5	
Marta	Vallès	45665432 morning 99999999 afternoon		+54875	Hope, 65	
Juan	Pérez López		juan@hot m.com	+34567		
Luisa	Fernández Martínez	935559876	luisa@ya hoo.com		Pine, 14	Tower Heights, 5

Anthropometric Data				
First name	Last name	Date	Weight	Height
Juan	Pérez López	15/5/09	120 kg	185 cm
Jusn	Pérez López	20/5/09	115 kg	
Luisa	Fernández Martínez	17/5/09	50 kg	160 cm
Luisa	Fernández Martínez	25/5/09	52 kg	1'60

Figure 2: A possible, valid state for Lalinia's MicroIS.

4 MICROIS MODEL

A MicroIS is basically a small data repository. However, its nature is closer to that of an office product than to the nature of a typical relational data base, since it has to manage sets of information of a reasonable size by means of an interactive tool that manages the structure, the data and the user interface altogether. This graphical interface will be based on a generic model for all the application although, as usual office tools do, will allow customization.

If we focus on the data perspective, the main difference among a DBIS and a spreadsheet is the independence of the stored data with respect to the scheme that determines its structure, as in the "Type Object" pattern by Johnson and Woolf (1997). This can be observed in the UML conceptual data model presented in Fig. 3, where there appear two areas neatly differentiated: data (to the left) and structure (to the right) of the MicroIS class.

Given that the data items exist independently of classifications, they will be identified individually by means of an artificial key generated by the system. Each item will have bound a set of properties (identified by their name inside the item, e.g., Last Name or Vacation Address) that will be composed by an ordered set of values (e.g., the class Telephone in those items that have more than one telephone). These values may be primitive values, references to other types or formulae.

At their turn, the type system defined by the schema of a MicroIS is composed of primitive types (supported natively by the system) and classes. Classes are generic definitions of items' structure (e.g., Personal Data and Anthropometric Data) and, as such, they define a set of attributes that will of a concrete type: primitive type, formula or reference (bidirectional, therefore it will be also required to know the inverse attribute, see for instance the *inverse* association in the figure). The classes allow adding consistency to the MicroIS whilst facilitating the input of data by the user. They also allow creating views, formula and queries since they provide a method for selecting items.

Each item may be classified as belonging to a particular schema's class. Each class will act as a template of the properties that are expected that the item has (each attribute will correspond with the property of the item that has the same name), in addition to establishing a set of validation rules (see Section 5).

5 SEMANTIC VALIDATION

Validations are part of the very schema and thus they are defined together with it, and its fulfilment is checked in the same way that the other constraints of the schema; in fact, the type of validations that a given item must fulfil will depend on its classes and attributes. A tool implementing the proposed paradigm should, consequently, validate the fulfilment of the schema and the validations for all the different items. The validation subsystem has been defined in an extensible way so that in future versions of the MicroIS concept, adding new types of integrity constraints becomes an easy process. Even, with an appropriate user interface, the ultimate intention is to let the final client of the particular MicroIS instance to define their own validation checks.

The validation subsystem has been integrated into the MicroIS conceptual data model, as it is shown in Fig. 4, where several restrictions currently implemented are shown.

It may be observed that the class Constraint offers two operations over Items. `Constraint::appliesTo?(i: Item)` indicates if a Constraint must be checked for a particular item, whilst `Constraint::compliant?(i: Item)` indicates if an item for which a Constraint must be

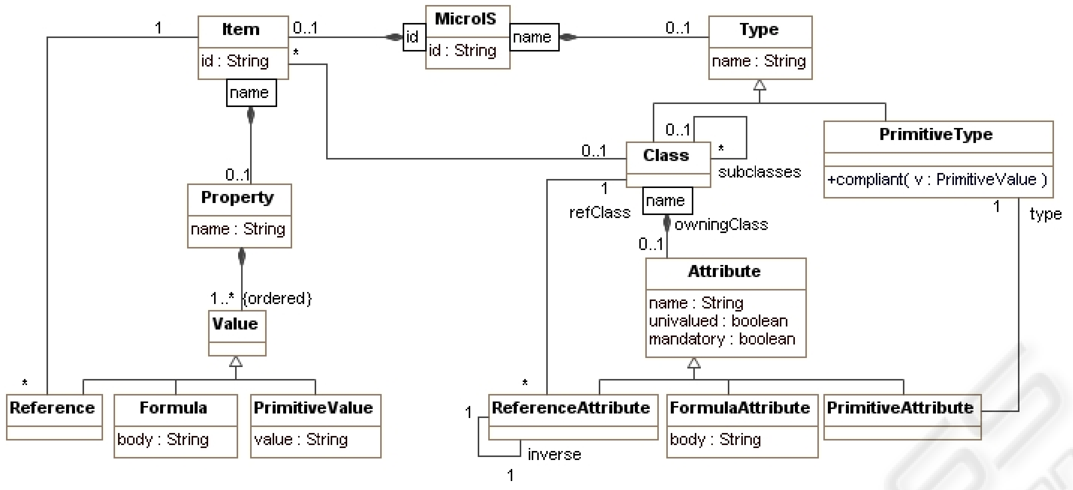

Figure 3: A UML data conceptual model for MicroIS.

check is compliant with it or else it violates it. These operations are abstracts, therefore they must be defined in each type of constraint. As an example, Fig. 5 shows the definition of those operations for two particular types of Constraint. Both constraints are applicable to items whose class corresponds to the class of the attribute bound to the Constraint.

6 TRANSFORMATIONS

From the viewpoint of data structure, the main characteristic of MicroIS is schema flexibility. To maintain the integrity of the information it is necessary to define exactly what transformations can be carried out on the schema and how they will affect both the scheme itself and the population of existing items. To study this problem we rely partly on Ambler and Sadalage's work (2006), which studies a set of refactoring operations for introducing changes in relational data bases that are already in production and proposes a series of techniques to gradually migrate systems utilizing these data bases. Unlike refactoring in data bases, modifying the schema is much easier when items are out of it and the compliance to the schema is optional and therefore it is permitted to be in a temporary situation where no all the restrictions are met.

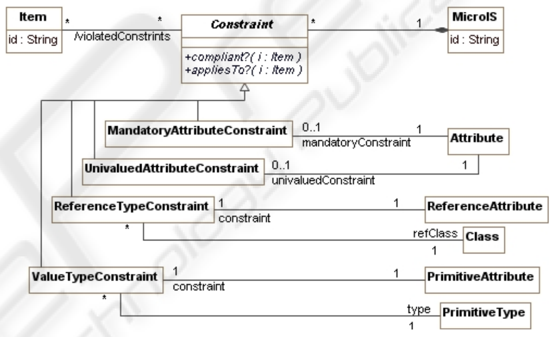

Figure 4: A UML data conceptual model for MicroIS validations.

```
context MandatoryAttributeConstraint::
appliesTo?(i:Item) =
i.class=self.attribute.class
context MandatoryAttributeConstraint::
compliant?(i:Item) =
i.property[self.attribute.name]->size()>0
context ValueTypeConstraint::appliesTo?(i:Item) =
i.class=self.primitiveAttribute.class
context ValueTypeConstraint::compliant?(i:Item) =
(i.property[self.primitiveAttribute.name]
->size()==0 or
i.property[self.primitiveAttribute.name]->
foreach(v|v.oclIsTypeOf(PrimitiveValue)
and
self.primitiveAttribute.
primitiveType.compliant?(v)))
```

Figure 5: OCL definition of two types of constraints.

So far, we have identified 16 atomic transformations that we have implemented in this first version of MicroIS. As an example, Figure 6 shows the detail of one of them, the classification of items, using a template that we have defined for this purpose.

Name: Classify Item

Summary: Binds a class C to an item I

Motivation: Indicates that the item I belongs to class C with the goal that, from this moment, all the validation rules defined over C are applied over I, and that I is included in all the views and queries of items of class

Effects over data: None

Effects over the schema: the item I includes the class C as its class

Restrictions that may be violated: All that apply over C

Contract: -- effects of the transformation on the model

```
context MicroIS::classifyItem
            (itemId: String, className: String)
  let I: Item = self.items[itemId] in
  let C: Class = self.types[className] in
  pre: I->notEmpty()
  pre: C->notEmpty()
  post: I.class = C
```

Figure 6: Definition of the transformation Classify.

7 CONCLUSIONS

This article has proposed a new paradigm of information system, the MicroIS, which stands in the middle between information systems based on databases and spreadsheets. This paradigm, although it had originated in the context of information systems for managing the core business of small organizations in a desktop application setting, could be applied in other type of environments. Concerning data, the main feature of the new paradigm is the ability to fluctuate between different states of consistency / structure according to user needs. For this purpose, it has been necessary to standardize and document all possible transformations on the scheme so that, at all times, integrity of information is preserved.

The paradigm that we propose involves the user in maintaining the integrity of the data to allow it to store information that does not follow the rules of integrity. For this reason we believe that this paradigm does not apply to systems where the information handled is critical (because of the possibility of inconsistencies) or systems where the volume of information is too large for manual processing and individual potential inconsistencies. Given our experience over 10 years in development projects for organizations of all types, conditions that make the MicroIS an attractive alternative are: flexibility, because the possibility of having different states of structuring and consistency, facilitates adaptation to changing requirements; low development cost, because it is possible to develop small information systems without the need for specialists and with an investment much lower than in the case of DBIS and with a return on investment much faster; usability, due to the existence of a generic interface for all MicroIS, which facilitates their usability.

Some aspects of MicroIS have not been covered in this article for space reasons, including self-structuring, the creation of queries (based on an expression language), the creation of views (based on the same expression language that queries) and the presentation of the generic user interface that allows working consistently with different MicroIS while establishing the standard mechanisms of personalization (based on views).

Our future work is aimed at giving more expressiveness to auto-modeling features, both regarding the detection of situations and the suggestion of alternatives, especially in reference to the identification of non-trivial situations like generalization / specialization relationships.

ACKNOWLEDGEMENTS

This work is partially supported by the project TIN2007-64753.

REFERENCES

Ambler, S. W., Sadalage, P. J. (2006). *Refactoring Databases: Evolutionary Database Design*. New York: Addison-Wesley.

Dabble blog. (2005). *Shouldn't be Spreadsheets*. Retrieved April 30, 2010, from http://blog.dabbledb.com/2005/03/shouldnt-be-spr.html

Baker, K. R., Foster-Johnson, L., Lawson, B. and Powell, S.G. (2006). *A Survey of MBA Spreadsheet Users*. Retrieved April 30, 2010, from http://mba.tuck.dartmouth.edu/spreadsheet/product_pubs.html.

Cunha, J., Saraiva, J. and Visser, J. (2009). *From Spreadsheets to Relational Databases and Back*. Proceedings of ACM SIGPLAN Workshop on Partial Evaluation and Program Manipulation, PEPM09, Savannah, Georgia, USA.

Johnson R. and Woolf B. (1997). *Type Object Pattern*. New York: Addison-Wesley.

Kraus, J. (2005). *The long Tail of Software. Millions of Markets of Dozens*. Retrieved April 30, 2010, from http://bnoopy.typepad.com/bnoopy/2005/03/the_long_tail_o.html.