Ministerul Educației Naționale
ROMÂNIA
UNIVERSITATEA DE MEDICINĂ, FARMACIE, ȘTIINȚE ȘI TEHNOLOGIE DIN TÂRGU MUREȘ

Scientific Bulletin of the „Petru Maior" University of Târgu Mureș
Vol. 15 (XXXII) no. 2, 2018
ISSN-L 1841-9267 (Print), ISSN 2285-438X (Online), ISSN 2286-3184 (CD-ROM)

QUALITATIVE METRICS FOR DEVELOPMENT TECHNOLOGIES EVALUATION IN DATABASE-DRIVEN INFORMATION SYSTEMS

Marius MUJI[1], Dorin BICĂ[2]
[1,2] *University of Medicine, Pharmacy, Sciences and Technology of Tîrgu Mureș*
Gheorghe Marinescu Street, no. 38, 540139 Tîrgu Mureș, Romania
[1]marius_muji@yahoo.com
[2]dorin.bica@ing.upm.ro

Abstract

Database-driven information systems are nowadays widespread in various application domains. The technologies employed to manage the database itself are generally built on the same data model – the relational model – and their evolution is mainly related to their physical performances, without significant changes in their conceptual foundation. By contrast, the user interface development technologies are changing at an increasing pace, which cause important problems related to learning costs and compatibilities with the existing systems. This paper provides a set of qualitative metrics which can be used to evaluate and compare different technologies, based on the most important conceptual objectives of database-driven information systems.

Key words: Database-driven application, information systems, declarative specifications

1. Introduction

Current technologies used to develop database-driven information systems can be classified in two distinct categories [1], [2]:

- The database management systems, used to build and maintain the integrated *community view* of the system;
- The development languages, used to develop the *user views* of the system, including all the user-specific data, presentation rules, and the necessary transformations needed to map every user view with the community view.

The database management systems are generally built around the relational model [3], which constitutes the common ground for the conceptual design of the community view, with all the advantages related to metadata compatibility and language standardization (e.g., the SQL standard [4]).

On the other hand, the development technologies used to specify the user views of the system don't share a common data model and require specific training for the application developers. Moreover, these technologies are changing at an increasing pace, which cause important problems related to learning costs and compatibilities with the existing systems.

This paper proposes a set of qualitative metrics, which can be used to compare different technologies, so that the developers can make the best choices, based on a consistent set of evaluation criteria.

The proposed evaluation metrics could become a valuable tool for the researchers of this field, by emphasizing the conceptual advantages of a certain research direction over another.

Section 2 introduces the conceptual foundation on which the metrics are conceived, Section 3 contains the definition of the proposed metrics, with an example of application, and Section 4 concludes by identifying the research directions revealed by the evaluation metrics.

2. Information systems' classical objectives

In this chapter we will discuss the objectives of the information systems, which were always important for the development community, and which explain the evolution of the technology in this field.

The first objective considered is related to the *structural specification*, through metadata, of the application domain's semantical constraints – as opposed to the *procedural approach*, where these constraints are expressed by the explicit specification of a sequence of CRUD (create, retrieve, update, delete) operations.

In the structural (data-oriented) approach, the information system's processes are specified as a sequence of transitions from a consistent state of the

database to another. In this case, the database management system would be able to (automatically) keep the database consistent. Thus, the information system can react with 'intelligence', doing nothing else than to preserve anytime the consistency of the database [5]. The entire semantic content of the system, including its "business rules" [6], [7], [8], [9], would then be specified through the system's metadata. Consequently, any change of the system requirements would determine some changes of the system's *declared* metadata, which would take the role of the information system's DNA [10].

By contrast, in a procedural approach, when the information system's procedures are defined as a sequence of CRUD operations, every specification of a system procedure should check if any of those CRUD operations violated the consistency of the entire system. Any change in system requirements would generate new sequences of procedural code, which usually have to be specified be the developer.

The first objective of the information systems can, thus, be expressed as *the possibility to formulate declarative specifications*, following a structural, data-oriented, approach to system development, as opposed to the procedural, process-oriented, approach.

The second objective identified is the possibility to accept new categories of users in the system, with minimal disturbances in the activity of the existing users. This objective's goal is to protect the existing users from the system specification changes, determined by the subsequent expansion of the system.

Under the ANSI-SPARC recommendations, first published in 1975 [2], [1], database systems address this objective through an architectural distinction between the integrated *community view* and the various *user views* of the system. The separation of the external level of the system from the conceptual level is meant to provide *logical data independence* – through which the database systems achieve the second objective formulated above.

The third objective considered is the possibility to change the technology without other changes related to the way in which data is defined and/or perceived by the users. This objective's goal is to be able to increase the physical performances (e.g., computing power, response time), without any additional development and/or operating costs.

Database systems address this objective, under the same ANSI-SPARC recommendations, through the architectural separation between the *logical specification* of the database (compatible with a certain logical data model) and its *physical implementation*. The degree in which the system can be specified at the logical level, without the need to access the physical level, would determine the degree of *physical data independence* ensured by the system.

The fourth objective considered is the possibility to define the system at *a high level of abstraction*, as close as possible to the natural language. This objective's goal is to reduce the development and maintenance costs, through improvements related mainly to the *learning curve* of the development technologies. Moreover, a higher level of abstraction in system definition implies a reduction of the *development time*, through the automatic generation of the physical specifications, based on the data models employed at the logical level.

An important challenge related to this objective is to achieve a certain level of abstraction at which the end users will be able to define the system themselves, using business-specific concepts and terms, without any other programming skills, determined by the implementation technologies.

3. Qualitative evaluation metrics for development technologies

The information systems objectives formulated in the previous section provide a valuable conceptual support for a set of qualitative evaluation metrics for current and futures development technologies.

Regarding the first objective, which refers to *the declarative specification of the system*, we considered two possible choices (the first being the desirable one):

1. The data-oriented (declarative) approach;
2. The process-oriented (procedural) approach.

Concerning the second objective formulated in Section 2, which refers to the *logical data independence*, we also considered two opposite choices (the first being the desirable one):

1. The *implicit* realization of the logical data independence, based on the meta-metadata compatibilities between the data model used to specify the user views, and the data model used to specify the community view, respectively;
2. The *explicit* realization of the logical data independence, through the complete specification of the necessary transformations/mappings between the data structures existent at the community level and those exposed at every user view level.

In all the cases with a low level of compatibility between the data structures employed for community view schema (e.g., the relations, in relational databases) and those employed for user view schemas (e.g. the collections of objects, in object-oriented technologies), there is a significant development effort to specify the necessary mappings between the corresponding data structures [11]. The usage of dedicated software tools, developed to overcome the so called "impedance mismatch" [12], [13] between the relational model and the object-oriented data abstractions, generically called ORMs (object-relational mapping) [14], [15], [16], generates significant development and maintenance costs, which could only be overcome through a higher compatibility at the data models' level.

For the third objective considered, related to the *physical data independence*, there are, also, two possibilities (the first being the desirable one):

1. *A high degree* of physical data independence, ensured by the abstract (mathematical) definition of

the data model employed at the logical level of the system;

2. *A low degree* of physical data independence, when a significant part of the information system cannot be defined at the logical level, and it needs some physical level specifications.

Considering the fourth objective, the one which refers to the abstraction level of the system's specifications, we considered four possibilities (in a descending order of desirability):

1. The system is specified at the *external level*;
2. The system is specified at the *conceptual level*;
3. The system is specified at the *logical level*;
4. The system is specified at the *physical level*.

The evaluation metrics defined above, synthesized in table 1, are extremely useful to compare different classes of technologies, like the relational database systems (RDBMSs), currently used to define the community view, and the object-oriented development languages, currently used to define the user views.

Table 1: The proposed qualitative metrics

Metrics	Possible options
1. Declarative specification of the system	*1. Declarative approach* *2. Procedural approach*
2. Logical data independence	*1. Implicit realization* *2. Explicit realization*
3. Physical data independence	*1. High degree* *2. Low degree*
4. Abstraction level for system specification	*1. External level* *2. Conceptual level* *3. Logical level* *4. Physical level*

If we consider the declarative/procedural metrics, we observe that, although current commercial implementations of the relational model do not support the entire declarative apparatus facilitated by the mathematical definition of the model [17], [18], they still provide declarative features, determined by the implementation of an essential data constructor (i.e., the relation), and the automatic enforcement of a predefined set of integrity constraints (e.g., the primary keys, the alternate keys, the foreign keys). Consequently, the relational systems are, essentially, declarative systems, due to their theoretical support, provided by the mathematical definition of their data structures, operators, and integrity constraints. The same conclusion could be drawn for any other class of technologies based on other data models, as long as the respective models provide the same kind of mathematical support as the relational model [19], [20].

Considering the object-oriented application development technologies, they usually provide some declarative support through certain design patterns that model typical business processes, but, in the general case, they need procedural specifications in order to model the behavior of the user interface (i.e., the presentation rules [5]) generated by every CRUD operation initiated by the end-user.

As we have already discussed, at the current stage of the technology, the logical data independence comes with the price of the explicit mappings between the community view's relational data, at one hand, and the collections of objects used to present the user views of the system, at the other hand.

On the contrary, using at the user views' level a data-model driven technology would provide implicit data structure mappings with the community data, provided that the data structures of the considered data model do have a native matching with the relational data.

Moreover, if the development technology is a data-model driven one [21], [22], the respective technology could provide a high degree a physical data independence, since the respective data model comes with an abstract, mathematical definition [19], [20]. That is not the case for the object-oriented technologies, which need, at some point, to 'brake' the encapsulation and to write some physical specifications, in order to define the system behavior determined by the CRUD operations initiated by the end-user [1].

Table 2 contains the results obtained by comparing, under the proposed evaluation metrics, the current object-oriented technologies with a data model driven technology, developed around a logical data model, derived from the relational model, and adapted for user view logical modeling.

Table 2: Data-model driven technologies vs. current technologies

	Data-model based technologies	**Current development languages (object-oriented)**
1. Declarative specification of the system	*Declarative (data-oriented) approach*	*Procedural (process-oriented) approach*
2. Logical data independence	*Implicit realization*	*Explicit realization*
3. Physical data independence	*High degree*	*Low Degree*
4. Abstraction level for system specification	*Logical level*	*Physical level*

From Table 2 it can be seen that a data-model driven approach would better satisfy the general

objectives of the information systems than current development languages.

4. Conclusions

Application development technology is changing so fast, that the professional community is struggling to keep up with its evolution. In this context, we need some well founded criteria which would support the developers to make the most appropriate choices, generating long term benefits related to technology switching cost and system maintenance.

Using four of the most important objectives of the information systems, which were always followed by the professionals of the field, we defined a set of qualitative metrics which can be used to evaluate and compare different development technologies.

Like it can be seen from the previous section, the proposed evaluation metrics can be used not only to compare two specific technologies, but to compare different classes of technologies, guiding the researchers of the field towards the best development directions. Thus, considering the application of the evaluation metrics presented in table 2, we can conclude that future research on development technologies should concentrate on data-model driven technologies, which would allow the declarative specification of the information system, would ensure the logical data independence at minimal costs, would ensure a high level of physical data independence, and, not least, would provide the logical support for high level system specifications.

References

[1] Date C. J. (2003), *An Introduction to Database Systems* (8th edition), Addison-Wesley.

[2] ANSI/X3/SPARC *Study Group on Data Base Management Systems*, Interim Report, FDT (bulletin of ACM SIGMOD), 1975.

[3] Codd E. F. (1980), *Data models in database management* in The 1980 workshop on data abstraction, databases and conceptual modeling, New York.

[4] International Organization for Standardization, ISO/IEC 9075-1:2008 (SQL/Framework), 2008.

[5] Date C. J. (2000), *What Not How: The Business Rules Approach to Application Development*, Addison-Wesley.

[6] Cemus K., Cerny T. (2016), *Automated extraction of business documentation in enterprise information systems ACM SIGAPP*, Applied Computing Review, vol. 16, no. 4, pp. 5-13.

[7] Kluza K., Nalepa G. J. (2017), *A method for generation and design of business processes with business rules*, Information and Software Technology, vol. 91, no. C, pp. 123-141.

[8] Ross R. G. (2013), *Business Rule Concepts*, Business Rule Solutions Inc.

[9] Halle B. v. (2001), *Business Rules Applied: Building Better Systems Using the Business Rules Approach*, Wiley.

[10] Steimann F., Kuhne T. (2005), *Coding for the Code*, Queue, vol. 3, no. 10, pp. 44-51.

[11] Bagheri H., Tang C., Sullivan K. (2017), *Automated Synthesis and Dynamic Analysis of Tradeoff Spaces for Object-Relational Mapping*, IEEE Transactions on Software Engineering, vol. 43, no. 2, pp. 145-163.

[12] Colley D., Stanier C. (2017), *Identifying New Directions in Database Performance Tuning*, Procedia Computer Science, vol. 121, no. C, pp. 260-265.

[13] Adya A., Blakeley J. A., Melnik S., Muralidhar S. (2007), *Anatomy of the ADO.NET entity framework*, ACM SIGMOD international conference on Management of data, Beijing, China.

[14] Rull G., Bernstein P. A., Santos I. G. dos,. Katsis Y, Melnik S., Teniente E. (2013), *Query containment in entity SQL*, ACM SIGMOD International Conference on Management of Data, New York, USA.

[15] Red Hat, November (2011) [Online]. Available: http://www.hibernate.org.

[16] Apache Software Foundation (2011), November 2011. [Online]. Available: http://cayenne.apache.org/.

[17] Date C. J., Darwen H. (2000), Foundation for Future Database Systems: The Third Manifesto (2nd Edition), Addison-Wesley.

[18] Haan L. d., Koppelaars T. (2007), *Applied Mathemathics for Database Professionals*, Apress.

[19] Muji M. (2013), *A Model for User Interface Design in Database-Driven Information Systems*, Scientific Bulletin of the Petru Maior University of Targu Mures, vol. 10, no. 1, pp. 24-27.

[20] Muji M. (2015), *Logical Operators for the Data-oriented Design of the User Interfaces*, Procedia Technology, vol. 19, pp. 810-815.

[21] Lewis B. (2010), *Data-Driven Molecular Specifications, Part 1*, The Data Administration Newsletter - TDAN.com, October 2010.

[22] Lewis B. (2010), *Data-Driven Molecular Specifications, Part 2*, The Data Administration Newsletter - TDAN.com, December 2010.