Perspectives on the Role of Business Rules in Database Design

Anca Ioana ANDREESCU, Marinela MIRCEA
Economic Informatics and Cybernetics Department, Academy of Economic Studies, Bucharest, ROMANIA
anca.andreescu@ase.ro, mmircea@ase.ro

Business rules are at the foundation of every information system as they drive and offer guidelines for managing and conducting all activities within an organizations. They are important both for operational systems and for analytical ones, as they can serve as data quality rules. Several business rules implementation solutions are presented and analyzed, according to some basic criteria that have to be taken into account when choosing a business rules implementation strategy. This paper also emphasizes how an explicit manipulation of business rules influence the database design.
Keywords: *database design, business rules, implementation strategies*

1 Introduction

Business rules approaches in software development are concentrated in finding ways and facilities that would support automatic propagation of business changes from business environment to software systems. This would help to bridge the gap between business and technology, as aligning information systems and business operation is one of the fundamental problems in all organizations [1]. To reach that goal, it has to be clear how business rules should be dealt with, as a special kind of software requirements, in each phase of the software development lifecycle. Accordingly, at least the following aspects have to be considered in this process: rules identification, business rules specification, rules implementation and rules management. A business rules implementation strategy assumes identifying both the place and the way of how to implement a business rule. The next sections of the paper will address several issues and challenges related to business rules, aiming to act as guidelines during business rules implementation. Particular emphasis will be given on the rules that govern the database model of an information system and various guidelines will be addressed in this regard.

2 Business rules overview

Over time, numerous studies and research projects aimed at the discovery, analysis, modeling, classification, formalizing and documenting business rules. It must be noted that in all researches, reference was made (explicit or not), on how the scientific approach relates to dimensions from the Zachman framework [2]. From this point of view, we have identified two research groups:

- which are based on a business vision, aiming at developing a business model, therefore on the activities conducted within the organization; remarkable here is the study Business Rules Team.
- which are based on a system vision, and therefore aim at modeling the information system; representative examples of such projects are : ESPRIT-I FIELD GUIDE Temporal or ESPRIT-II.

As part of a consortium of 17 organizations, known as "The Business Rules Team", Business Rules Group (BRG) participated in the proposal from the Object Management Group (OMG) to analyze the semantics of business rules from a business perspective. The answer of BRT, called Semantics of Business Vocabulary and Business Rules - SBVR, is a comprehensive study on the semantics of business vocabulary and business rules, which have considered the following aspects [3]:

- The focus on business, in that there's no reference to information systems or concepts of information systems - focuses exclusively on how business people think.
- Business rules are fully specified from business vocabulary.
- Semantics integrity rules should be based on predicate logic.
- People involved in the business (as "customers") communicate vocabulary and business rules to computer professionals (as "suppliers"), in the form of system requirements specifications.

Regardless the perspectives from which they are being analyzed, business rules always belong to an organization and must occur in such a way that is acceptable to the business [4].

Graham Witt [5] distinguishes two main types of rules: a) operative rules, which describes behavior and state what must or must not happen in particular circumstance. They include the so-called "if-then-else" rules or event-action rules; and b) definitional rules which constrain the definition of various constructs created by the organization. These may be simple business facts, derivation rules, complex decision table rules etc. Additional, the author mentions several ways in which rules may be applied in an organization:

- rules governing the business processes performed by the organization through their employees, customers, suppliers and partners;
- rules governing user interfaces and electronic messages;
- rules ensuring database integrity;
- rules governing human behavior other that business processes.

Because some of the business processes may be automated or not (or partially automated) we can conclude that rules related to user interfaces, electronic messages and databases are likely to be fully implemented in the information system.

3 Business rules implementation solutions

Last decade was marked by the development of a wide variety of technologies, instruments and software products that support the implementation and maintenance of business rules. This section presents the main conclusions of an ample study [6] regarding a large set of possibilities to translate business rules from the business domain to the technology domain. The considered solutions will be gradually presented, starting from those who offer a minimum support for business rules management and getting to those who offer the premises for an efficient rules management. Likewise, a wide area of technical solutions was covered.

Probably the most common way to implement a business rule is to mingle it inside de application source code. In this case, it is advisable to use some style conventions in order to mark the source code parts that implements business rules [7]. Subprograms and program statements separated by user defined limits are a first step towards emphasizing and encapsulating business rules.

The specific constructors of the Design by Contract (DbC) method (pre-conditions, post-conditions and invariants), advocated by Berthrand Meyer in [8], are suitable for implementing business rules, allowing rules identification in the source code and increasing software reliability. Starting from this remark, in [9] and [10] we have proposed and illustrated various ways of DbC-like rules specification and implementation using: a) formal languages (Object Constraint Language), b) programming languages that supports DbC (Eiffel, XC#) and c) a business rules extension of the „Contract" design pattern [11].

Aspect oriented programming intends to complete object oriented programming (or other paradigms), by extending the possibility to modularize applications. As

aspect allows separating and reusing certain software requirements, they are a good candidate for implementing business rules, as stated and illustrated in [12].
Another step forward in rules modularization is the separation of rules in files that contains source code written in a script language, like VB Script. Choosing this implementation solution will have to take into account the low application performance, due to the fact that, for interpreted languages the code runs rather slow.
The existence of software components as stand-alone entities represents a good premise for encapsulating decisions based on business rules into such components. Their main advantage is due to the possibility to modify a rule implementation without affecting the base applications that reference the component. Rule components are seldom provided from external sources, because they implement a part of the application business logic, and therefore, are not very general.
Business rule markup languages play an important role within the Web environment, as they allow the specification of business rules as modular and independent units, in a declarative manner, as well as rules publication and interchange between different systems or tools. The proposal for the standardization of the RuleML [13] language proves an intense work in amending rule markup languages, despite their lack of usage in practice.
Business rules engines are a software technology that guerdons the efforts to identify efficient business rules implementation solutions, as they were designed with the exclusive scope of rules administration. As opposed to business rules components, a rule engine do not solve a particular problem, but provides a set o generic capabilities to define, store and apply business rules. But, above all things, a rule engine must manage the dependences between rules, using inference mechanisms. It can be used as a standalone toll, without storing business rules in a dedicated repository (like Jess or Drools), or it can be part of a special type of software system, named Business Rules Management System (like IBM JRules).
Very often business rules are associated with the control of business behavior. Research and practical solutions in the field of business process management and automation, have dealt with the problem of business rules implementation. Therefore, most of the process engines are able to reference other components that implement rules, while the rules that are internally implemented within a process engine will be most probably represented in a proprietary format, as part of the product that automates the business processes (for example, Microsoft Windows Workflow Foundation [14].
Rules that influence or constraint business objects will be, most probably, included in the system's database, from where they have the fastest access to data. Database mechanisms are able to implement business rules as constraints, stored procedures and triggers.
Table 1 depicts a comparative analysis of the implementations solutions presented above, by emphasizing their main advantages and disadvantages with respect to the following criteria: (S) rules are separated by other implementation aspects, but they are still placed in the same file as the rest of the code. By dignifying rules position in the source code, they can be easily located. (E) rules are externalized and separated from the system's business logic. (L) rules are expressed in a high level specification language, suitable for the non-business people. (R) relations among rules can be determined, similar to inference mechanisms. (D) the solution is dependent of a software producer.

Table 1. Comparative analysis of the business rules implementation solutions

SOLUTION	**CRITERIA**					**ADVANTAGES**	**DISAVDANTAGES**
	S	E	L	R	D		
Subprograms and program statements	x					• ease of integration in code • high execution performances	• a rule change implies base code modification
Design by Contract constructions	x					• rules are separated by subprograms • testing facilities	• a rule modification implies recompiling the base application
Aspect Oriented Programming		x				• rules are placed in files external to the base code	• debugging problems
Script languages		x				• rules are placed in files external to the base code	• low execution performance
Rule components		x				• a rule modification does not imply recompiling the base application	• low generality, due to the method parameters from the interface
Business Rules engines		x		x	x	• rule reuse in multiple system points • allows the management of complex set of rules	• rules tend to concentrate in one place
Business Rule markup languages		x				• rules can be interchanged • allows platform interoperability	• few tool support and practical applications
Database mechanisms		x				• high application performance (Constraints and Store procedures) • insure consistence along all system modules	• low application performance (Triggers)
Business process engines		x		x	x	• separate business rules from business processes	• rule representation in a proprietary format

SOLUTION	CRITERIA					ADVANTAGES	DISAVDANTAGES
Business rules management systems		x	x	x	x	• increased rule management facilities • allow business people to define and modify rules	• significant financial investment • implies additional security issues

It is highly probably that, for a certain software system, a business rules implementation strategy will encompass a combination of different approaches discussed in this section.
The decision to choose one implementation technology is influenced by many factors among which the most important is the type of system which is being developed. Through a simple analysis, we have identified four main ways to implement rules, as described in Figure 1. Assuming that every software systems stores the information about business objects within a database, the hypothesis stating that some of the business rules will be implemented at the database level is plausible, mostly due to performance reasons, but also to force all the system's applications which use the database, to follow the same business rules. Starting from the above observations, in Figure 1, is presented a pattern that establish the link between the type of software system and the ways to implement rules.

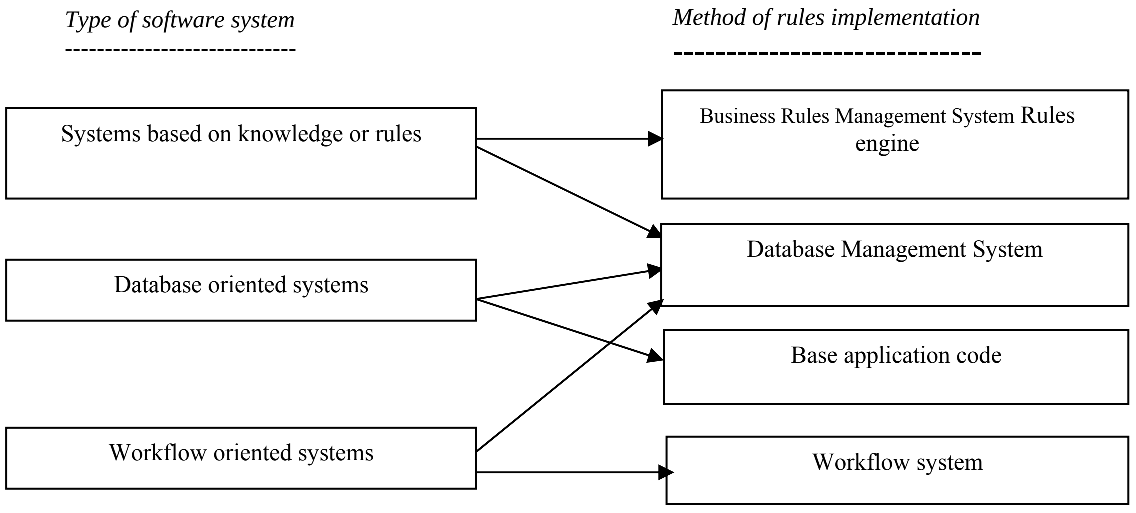
Type of software system
Method of rules implementation
Systems based on knowledge or rules
Database oriented systems
Workflow oriented systems
Business Rules Management System Rules engine
Database Management System
Base application code
Workflow system

Fig.1. Correspondence between the type of software system and the business rules implementation methods

The great variety of implementation solutions of business rules leads to the necessity for a well founded and competent analysis of the system's needs. Therefore, at least three software quality characteristics must be explored: efficiency, maintainability and reliability. Efficiency will be evaluated with respect to the final user who demands a quick answer from the system in such a way that the business rules must be verified and executed efficiently. Maintainability resides in the degree of easiness necessary to update the application in order to continue using it even under modified conditions. From the business perspective, the system must provide facilities for: business rules specification, adding new rules, deleting those who cease to operate, modifying existing ones or restructuring relationships between rules. Reliability has to be analyzed especially from the perspective of correctness, meaning the degree in which the system fulfills intended functions and imposes the business rules required by the beneficiary.

4 Business rules in databases

As mentioned before, besides rules that affect business processes, there are also rules that apply to the data. Rules that affect data will fit, most likely, in the database system, a place where they will be best positioned for access to data. Database technology that is currently most commonly used is the relational model, represented by products such as Oracle or Microsoft SQL Server, so we build the discussion around them. Relational Database Management Systems (RDMS) as the one referred to above, do more than store and collect information: it offers mechanisms capable of controlling the characteristics of application data [15].

In the following, we analyze three types of database features that can be used to implement rules: constraints, stored procedures and triggers.

A. Constraint is a generic term related to table columns which limits data values in order to preserve database integrity. Integrity of the relational model restrictions are structural and behavioral [16]. **Structural constraints** include:

1) Unique key restriction: a table should not have multiple tuples with the same value for all keys. This restriction is provided by a RDMS by not allowing duplicate values for a primary key field.

2) Entity restriction: for a table, the primary key attributes must not take NULL values. This restriction is provided by a RDMS also by not allowing null values for a primary key field.

3) Referential restriction: in a table t1that reference a table t2, foreign key values must include the primary key values of t2 or the value NULL (unspecified);

Database design should be influenced on the meaning that business people and business analysts give to certain objects or terms from the business vocabulary. And discussions regarding these aspects might be endless. If we consider a very simple e-commerce application, we must define who a client is: a) the one who visit the web site; b) the one who places an order not being authenticated; c) the one who places an order being authenticated; d) or the one who finalize an order being authenticated. Depending on these different interpretations, the database design could slightly differ. For the more restricted scenario of case c), in business rules modeling, such a situation is expressed in the form of a simple fact: A Client can place one or more Orders. One way to enforce this when entering a new order in the database is to specify the NOT NULL property for Client column. This means we have provided a value for the client to create an adequate record. An alternative to referential integrity is to allow NULL values for reference (case b) from the above example). This would allow an object Order to be created and initially disassociated and an association to be added later. Since both scenarios are plausible, their associated business rules will determine which of them is correct in a particular case.

Behavioural constraints are those which are defined by the behaviour of data and take account of existing values in relational database. By restricting the field, it can be checked whether the corresponding attribute in a table is between certain values or whether it conforms to a prescribed format. The relationship between columns of the same table can also be. Behaviour constraints being very general are managed either when data description (e.g. CHECK clause) or outside the model at runtime. We present two examples of constraints that apply to Customer and Order tables, using SQL Server specific syntax.

Rule 01: A customer identification code is defined as a four-digit integer.

```
ALTER TABLE Client
ADD CONSTRAINT
BC_R01_ClientFormat
    CHECK
    (IDClient LIKE ' [0-9] [0-9] [0-9] [0-
9]')
```

Rule 02: An order is, by definition, in one of the following states: Pending, Approved, Finalized and Delivered.

```
ALTER TABLE Order
ADD CONSTRAINT
BC_R22_OrderState
    CHECK
(state = ‘Pending’ OR state = ‘Approved’
OR state = ‘Finalized’ OR state =
‘Delivered’)
```

Incorporation of control mechanisms in a database prevents the occurrence of any actions that contravene the rules, not only when the object is created, but also for any future attempts to change it. The major disadvantage of these constraints is that they simply can’t be used for more complex conditions, involving several objects, resulting in association of multiple tables in a relational database.

As with other solutions that implement rules, it should not be forgotten that we will need to find a way to determine where the rule has been codified. For example, you can adopt a naming convention rules, as in the examples above, which can be used to track or to extract rules from the system [7].

B. As their name suggests, **stored procedures** are procedural code modules compiled and stored in the database (as any object manipulated by a RDMS) without having to be screened a second time to execute [17]. Compared to any other source code, it has some advantages. First, because they are so close to the data, can be more efficient to implement data-centric rules. Also, for distributed systems where there are several applications that share the same database, stored procedures can be a guarantee that the rules contained therein are used consistently within each application. The same is true for newly created applications that will not have to re-implement the rules that apply to the entire system.

C. A **trigger** is a special type of stored procedure that is not invoked explicitly, but is triggered automatically when a certain condition is detected. Triggers are attached to an event within a database table. The event can be an action to add, modify or delete entries in the table or a combination of these actions using logical OR operator.

Although is triggered by an event occurring in a table, a trigger is not limited to the characteristics of that table. Unlike simple constraints discussed above, a trigger can be used to implement constraints in a table, between the tables, between databases and even between servers. However, these advanced capabilities require taking some compromises. Triggers are slower than other approaches, such as simple constraints. It must also be taken into account the danger that inexperienced users can do when using these facilities ineffective, because overall system operation can have strong negative effects. Therefore, some authors even recommend avoiding the use of triggers, at least when we are not sure of the effects they can generate.

5 Conclusions

The construction of a rules repository and the specification of business rules in a structured and/or formalized manner constitute important steps towards raising the quality of a software system that treats rules as explicit requirements.

As guideline for developing a database model it is very important to use a standardize vocabulary because business stakeholders will comprehend the meaning of a rule more precisely if those specifications use business terminology rather than the names of database tables or columns. Experience shows that, despite the fact that all concepts seems to save a well-defined meaning inside the organization, many business terms turn out to have several meaning depending on the context and the person who uses them. In this regard, we have presented in [18] a set of business rules patterns for rules specification.

Concerning efficiency and maintainability, these must not be analyzed separately, as between they exist certain interdependent relationships. The end result is that the evaluation of implementation strategies must balance the efficiency and maintainability requirements of the system.

Acknowledgements

This work was supported by CNCSIS-UEFISCSU, project PN II-RU (PD), “Modern Approaches in Business Intelligence Systems Development for Service-Oriented Organizations Management”, code 654/2010, contract no. 12/03.08.2010.

References

[1] M. Bajek, M. Krisper, Issues and Challenges in Business Rule-Based Information Systems Development, Conference on Information Systems, 2005.

[2] J.A. Zachman, A framework for Information Systems Architecture, IBM Systems Journal, Vol. 26, No. 3, 1987.

[3] Semantics of Business Vocabulary and Business Rules, available at: http://www.omg.org/docs/dtc/06-03-02.pdf

[4] B. Von Halle, Business Rules Applied, John Wiley & Sons, 2002.

[5] G. Witt, Writing Effective Business Rules, Morgan Kaufmann, 2012.

[6] A. Andreescu, A. Uta, R. Mihalca, Business Rules Implementation Strategies, The Ninth International Conference on Informatics in Economy, Bucharest, ASE Publishing House, INFOREC Publishing House, 2009.

[7] T. Morgan, Business Rules and Information Systems: Aligning IT with Business Goals, Addison Wesley, 2002.

[8] B. Meyer, Object-Oriented Software Construction, Second Edition, Prentice Hall Professional Technical Reference, Santa Barbara,1997.

[9] A. Andronescu, How to Build Reliable Business Applications with Design by Contract, Seventh International Conference on Informatics in Economy, ASE Publishing House, INFOREC Publishing House, Bucharest, 2005.

[10] A. Andreescu, Dezvoltarea sistemelor software pentru managementul afacerilor, ASE Printing house, 2010.

[11] M. De Champlain, The Contract Pattern, PLoP ’97 Conference, Monticello, Illinois, SUA, September 1997.

[12] M. A. Cibrán, Connecting High-Level Business Rules with Object-Oriented Applications: An approach using Aspect-Oriented Programming and Model-Driven Engineering, PhD Thesis, Vrije University, Brussel, June 2007.

[13] DATA H. Boley, The RuleML family of web rule languages, the fourth workshop “Principles and practice of semantic web reasoning”, Budva, Montenegru, June 2006 .

[14] D. Chappell, Introducing Windows Workflow Foundation, Microsoft Corporation, 2007, available at: www.davidchappell.com/IntroducingWCFv1.2.1.pdf.

[15] C. Strîmbei, Modalități de organizare a datelor în sistemele informaționale economice, PhD Thesis, “Alexandru Ioan Cuza” University, Iaşi, 2005.

[16] I. Lungu, Baze de date oracle limbajul SQL, ASE Printing house, Bucureşti, 2005.

[17] I. Popescu, Prelucrarea avansată a informației.Oracle 8, Tehnică Pronting House, Bucureşti, 1999.

[18] A. Andreescu, M. Mircea, Managing Knowledge as Business Rules, Informatica Economica Journal, nr.4/2009.

Anca Ioana ANDREESCU is university lecturer in Economic Informatics and Cybernetics Department, Academy of Economic Studies of Bucharest. She published over 15 articles in journals and magazines in computer science, informatics and business management fields, over 20 papers presented at national and international conferences, symposiums and workshops and she was member in over nine research projects. In January 2009, she finished the doctoral stage, the title of her PhD thesis being: The Development of Software Systems for Business Management. Her interest domains related to computer science are: business rules approaches, business analytics, requirements engineering and software development methodologies.

Marinela MIRCEA received her degree on Informatics in Economy from the Academy of Economic Studies, Bucharest in 2003 and his doctoral degree in economics in 2009. Since 2003 she is teaching in Academy of Economic Studies from Bucharest, at Informatics in Economy Department. Her work focuses on the programming, information system, business management and Business Intelligence. She published over 25 articles in journals and magazines in computer science, informatics and business management fields, over 20 papers presented at national and international conferences, symposiums and workshops and she was member over 15 research projects. She is the author of one book and she is coauthor of four books. In February 2009, she finished the doctoral stage, and her PhD thesis has the title Business management in digital economy.