ISSN 1392 – 124X INFORMATION TECHNOLOGY AND CONTROL, 2007, Vol. 36, No. 3

BUSINESS RULES MANIPULATION MODEL[1]

Liudas Motiejūnas, Rimantas Butleris

Kaunas University of Technology
Studentų St. 50, LT51368 Kaunas, Lithuania

Abstract. Business rules approach is quite new and oriented at software systems in which the rules are separated, logically and physically, from other aspects of the system. In this paper some of the methods for description of the business are discussed. Business rules activation from database triggers is presented. Implemented prototype and algorithms are discussed.

1. Introduction

Business rules are precise statements that describe, constrain and control the structure, operations and the strategy of a business. They can be found everywhere in a raw, unstructured form. The business rules are the most changing part of the business. We can describe a business rule as "a statement that defines or constrains some aspect of the business. It is intended to assert business structure or to control or influence the behavior of the business" [9].

1.1. Describing business rules

There are a few main ways for describing business rules. One of them is to store rules in objects. Five design principles for checking business rules in objects are presented below [12]:

1. Business rule checking occurs when objects change state.
2. The object changing its state invokes business rule checking.
3. Business rule checking invokes logic that may or may not reside in the affected object's implementation.
4. Business rule checking must be flexible enough to allow selective bypassing by trusted services.
5. Business rule checking is separate from logic rule checking.

However, business rules are not independent and they belong to objects. Business rules should be associated to entities and events. Business rules should be kept in separate business rules repository and managed separately from applications' code.

Other way is to implement business rules as database triggers. There is no united opinion about using the data base triggers for the implementation of the business rules. Some authors state that database triggers are used to implement certain types of business rules, such as facts, constraints, actions, enablers and derivations [8]. Other opinions declare that business rules have absolutely nothing to do with programming, database triggers or expert systems [2]. In our opinion, business rules should not be implemented as database triggers and must be kept in business rules repository separately in a declarative form.

Another way of describing business rules is OCL. Often simple rules in plain English require several lines of OCL code to represent [1]. Trying to use OCL to describe a business rule written in several paragraphs of plain English text would be extremely complex. OCL is based on first-order predicate logic [18]. Business people would hardly understand this kind of language. Anyway, OCL can be used for both: writing assertions and describing business rules. However, for OCL to act as a language for writing business rules, it would have to be the perfect syntax for all the business situations, but we cannot unambiguously say that this is true. OCL should concentrate on the primary purpose, and get that right, before trying to be general-purpose business rule syntax.

Rule mark-up languages will be the vehicles for using rules on the Web and in distributed systems. They allow deploying, executing, publishing and communicating rules on the Web [17]. They are also converging towards a lingua franca for exchanging rules between different systems and tools. In a narrow sense, a rule mark-up language is concrete (XML-based) rule syntax for the Web. In a broader sense, it

[1] The work is supported by Lithuanian State Science and Studies Foundation according to High Technology Development Program Project "VeTIS" (Reg. No. B-07042)

should have an abstract syntax as a common basis for defining various concrete sublanguages that serve different purposes. The goal of RuleML is to permit reusability and interchange at a higher level. Anyway, RuleML raises similar problems like XML: it requires an investment from the rule engine vendors to implement, and it requires an additional investment from the software developers and architects to add disambiguating information in the condition of the rules to guarantee the expected behavior.

1.2. Business rules repository

Physically, rule repository is an autonomic business rule collection which can be altered at any time using relatively easy tools [4]. These are two solutions to storing rules:

1. Parameter driven approach. In this case rules are stored in the database where they are characterized by the values of various attributes. It has been shown by different researchers that rule repository could be designed as an independent database [14] or as a part of the main logical model [13]. However, the first solution offers more flexibility and more options for the storage of complicated business rules.
2. Independent process-driven approach. This approach is similar to the traditional methodologies where rules are implemented directly in the program code, only in this case the code, representing rules, is stored independently from other layers of the IS and therefore rules are expressed only once in the system.

Business rules repository is a database that stores all the data about business rules and all the necessary metadata about entities, attributes and relationships that are included in the data model. Probably it is impossible to create an universally accepted business rules repository structure, because various organizations or business rules researchers define different types of business rules and ways that the rules are described [3, 10, 15]. In our case we use the first solution to store business rules.

1.3. Business rules engines

The system-wide enforcement of stored rules is managed by the special rule interpretation mechanism called business rules engine. Such an engine is considered as a monolithic mechanism [11], however, the task of enforcing or implementing rules can be carried out by more or less independent services.

The business rules engine calls business rules from business rules repository and performs actions described by the rule. Business rules engine can be implemented in various ways [16, 19], the same as business rules repository, because its architecture depends on business rules repository (the form that business rules are stored in it). A wider overview of the business rules engines is presented in [7]. It doesn't matter in which way the business rules engine is implemented - it must ensure that business rules are performed correctly.

2. Business Rules Repository model

We used business rules repository which is based on repository model introduced by Plotkin [14]. This business rules repository is flexible and easily extendable; it gives a possibility to store various types of the business rules. The main improvement of Plotkin's repository is metadata about logical data structure incorporation with repository. Connecting business rules with data through metadata makes rules more independent. Next improvement of the repository is possibility to store functions which are performed by the business rules. Actions that are needed to perform these functions are also stored in the repository. This business rules repository is presented in [5, 6].

3. Activation of Business Rules

Because business rules are kept in the business rules repository separate from the program code, they are declarative and implicate no control logic, they have to be called and executed by business rules engine. Every rule rejects, produces or projects some type of actions or data. Also each rule is associated with particular data. Until the user does not take any action, business rules are not called from the business rules repository. But when some action occurs, the business rules engine must verify business rules and evaluate that action. Generally, the business rules engine starts working on the three basic events – INSERT, DELETE and UPDATE. Business rules that respond these events are similar to data base triggers; however they are kept separately from the database. Either way when the user makes an attempt to insert, delete or update data, the business rules engine must fire business rules that are associated with data that the user wants to change. It is like a monitoring process – business rules engine is inactive and comes to action only when some changes in data are noticed. Business rules of this type must be fired at least on two separate events.

But there can be other kinds of business rules in a system. That means, business rules are fired in other situations, not only when the user attempts to change data. This kind of rules is not associated with the data control; they can create data themselves. As an example, business rules of SELECT or CALCULATION type could be pointed out. The result of such business rules is derivative data that can be stored in a file, shown on the screen or printed in a report. These rules usually are called by specific events, which can depend on user's actions (button click) or simply on the timer. Business rules of this type do not have to be fired at least on two events, because they do not ensure the consistency of data, they can create data themselves.

4. Invoking Business Rules from Database Triggers

One of the main problems in business rules systems is to activate business rules and bring them to life. "Data control" business rules serve the same purpose as database triggers. These rules are responsible for data that are kept in the database. They even react to the same events as database triggers. Business rules differ from database triggers, because they reside in business rules repository and they must be activated by some particular action. Usually it is difficult to capture different events that can activate business rule. In the following example database triggers are used because reaction to events INSERT, DELETE and UPDATE is already implemented. We only need to extend a database trigger with appropriate line that activates business rules engine:

```
CREATE TRIGGER [Trigger_Name]
   ON Table_Name
FOR {[INSERT] [,] [UPDATE] [,]
  [DELETE]} AS
BEGIN
  declare
@tablename int,
@rowsAffected int,
@nullRows int,
@validRows int,
  select @rowsAffected = @@rowcount
call Event(tablename, rowsAffected,
     nullRows, validRows, rowcount )
End
Call callengine(temp_table)
GO
```

5. Invoking Business Rules by Specific Events

Some business rules can be activated by specific situations in business systems. These situations depend on user's actions and are not related with data control. Such business rules cannot be activated from database triggers, but by appropriate events occurring in a business application (Click(), ValueChange(), etc.). Specific code line is needed which activates business rules engine in the application:

```
CallEngineSE(Event, F_Action_ID)
```

Formally we can describe requirements for the business rules activating as follows:

1. Each rule must have an event that activates it:

$$\forall x (Rule(x)) \rightarrow \exists z (Event(z) \cap Activates(z, x))$$

2. Each rule must have a related function of action:

$$\forall x (Rule(x)) \rightarrow \exists y (F\_Action\_ID(y) \cap Related(y, x))$$

The third requirement is mandatory only for "data control" business rules that are activated by the database triggers.

3. Each rule that is activated from database triggers must have a related table:

$$\forall x (Rule(x)) \rightarrow \exists z (Table(z) \cap Related(z,x))$$

6. Business Rule Processing

A number of actions is performed in the business rules system when some user-defined action occurs. Actions have to be performed in an appropriate priority. The sequence diagram showing these actions is presented in Figure 1.

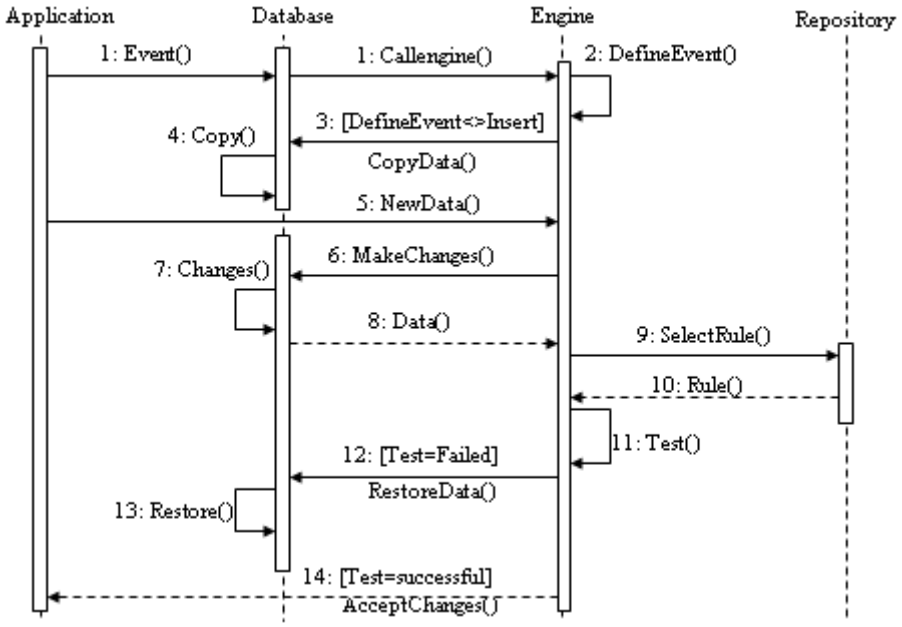

Figure 1. Sequence diagram of business rule processing

When a user attempts to change the data, database trigger detects an event and calls a business rules engine. The Business rules engine defines what kind of event it was. For event INSERT existing data is not

made copied, for other two events (DELETE, UPDATE) data must be copied. Then user-defined actions (changes to existing data) are accepted and business rules engine selects appropriate business rule from business rule repository according to given parameters. The test described by the rule is performed. If the test fails, the initial data are restored, otherwise changes to data are accepted.

7. Business rules engine implementation

During research we assume that business rule set used for experiment is logically correct and we do not analyze data specification. Abstract service ordering process was implemented for testing described algorithms. The set consisting of 15 business rules was entered into business rules repository in order to check possibility to enter different types of rules. The fullness of the business rules repository was checked using rules classification presented by GUIDE Project.

All implemented functions and procedures are shown in Figure 2. The structure of the code matches diagram hierarchy shown in this figure.

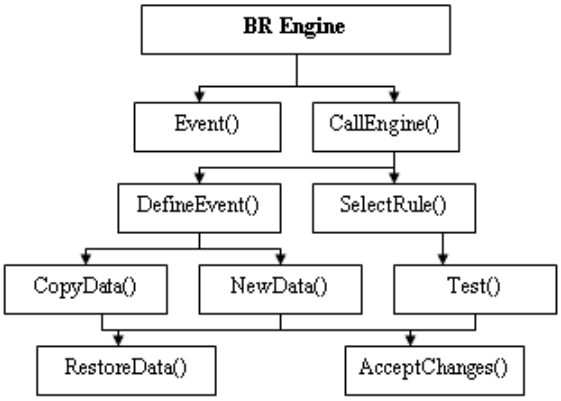

Figure 2. Function hierarchy diagram

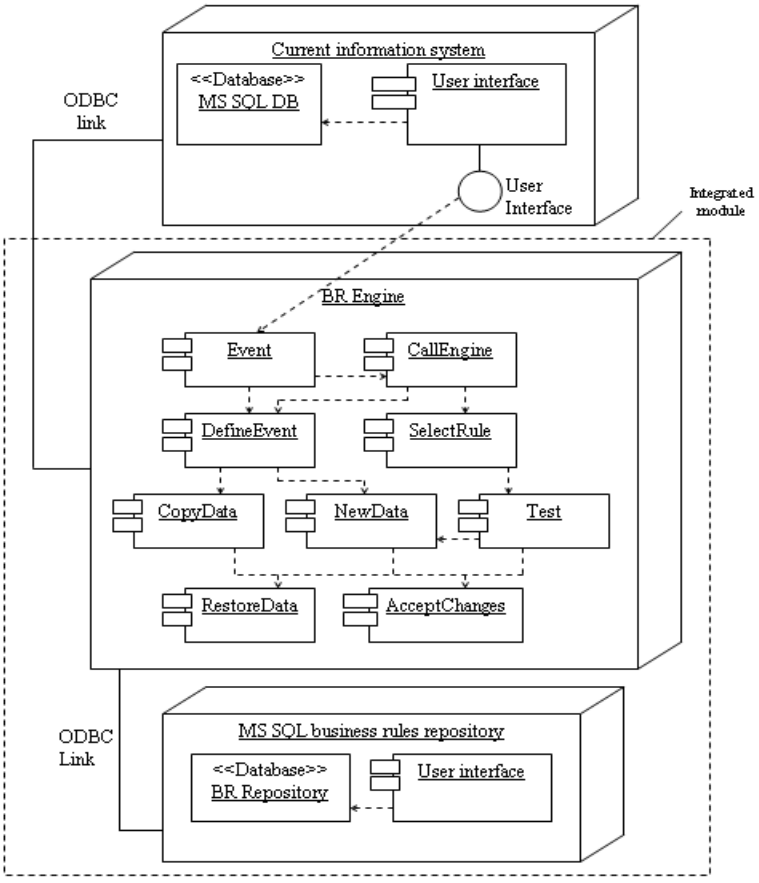

Figure 3. Architecture of the developed prototype integration into information system

Functions and procedures are executed according to the hierarchical structure of the given diagram. Two procedures and five functions were implemented during experiment.

A detailed architecture of the developed prototype integration into an information system is shown in Figure 3.

Algorithms of the two main procedures in the business rules engine are shown in Figure 4 and Figure 5.

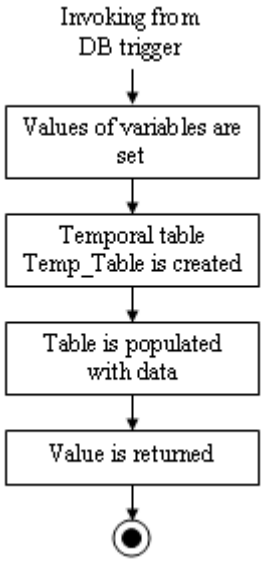

Figure 4. Algorithm of the procedure Event()

After invoking procedure Event() from database trigger in the first step the values of variables are determined:

- Table name – @tablename;
- Column ID – @rowsAffected;
- Event ID – Insert, Update, Delete;
- Changed record attribute – @nullRows, @validRows;
- Changed record count of elements - @rowCount.

In the second step temporal table TEMP_TABLE is created, which consists of the following columns: Row_ID, Change (mark is set if the field was changed), Event_ID (common for all fields), Range_Nr (matches predicted number of cycles). In the next step appropriate data are set by the given fields and table TEMP_TABLE is populated. If this table was populated succsesfully, then value TRUE is returned.

The algorithm of the procedure CallEngine() is shown in Figure 5.

In the next step of the experiment abstract service ordering process was implemented, but only in a traditional way – business rules were buried in the code.

Table 1 shows the difference between numbers of code lines, when business rules are implemented in traditional way and using the proposed business rules engine model.

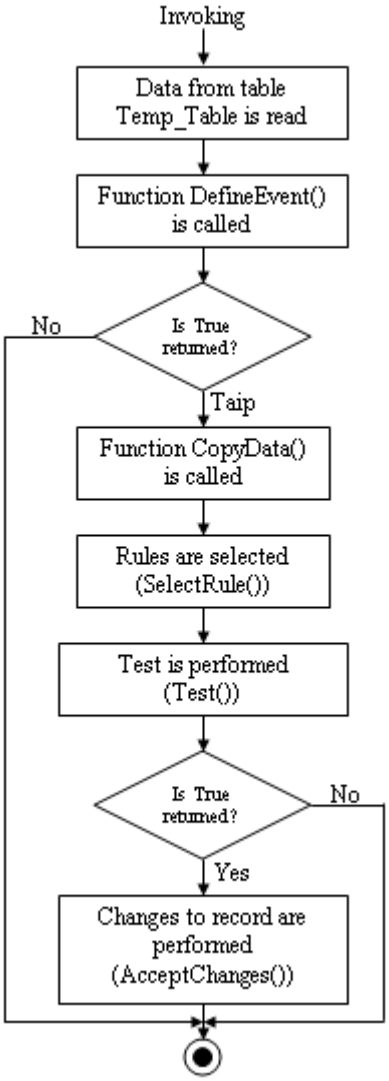

Figure 5. Algorithm of the procedure CallEngine()

As the table shows, number of code lines needed for different rules can vary a lot. However, storing business rules in the repository and using proposed business rules engine, there is significant reduction of work that is needed to accomplish the same tasks. But it has to mentioned that the designer or programmer will have to enter business rules into rules repository, therefore reduction of work won't be as significant as it is shown in table 1.

8. Conclusions

Metadata are already used in different DBMS. Resumptive metadata storage model is independent of particular database management system. Metadata allow navigating in the logical database without any additional programmed operations and identifying data elements that are impacted by particular business rules. Relating business rules with data not directly, but through metadata, better business rules independence is achieved.

Table 1. Difference between numbers of code lines

Business rule	Number of code lines	
	Coding	**Using BR**
Client must have last name, first name, address and age	11	7
Client type must be only one of the following: Golden, Silver, Usual	4	
Age of the client must be greater than 18	2	
Additional actions	*16*	
Order can be entered only by manager	6	6
If client ordered services for more than 7.000lt, then client type is Golden	19	
Additional actions	2	
Total number:	**60**	**13**

The proposed business rules engine model gives us several advantages. Here are the main ones:

1. Differently from systems where rules are implemented as database triggers, in this work triggers are used only to invoke business rules. That's why during business rules implementation there is no need to have specific programming skills.
2. There is no need to indicate particular business rules, which have to be managed, because according to passed parameters needed business rule or rules set is invoked depending on situation.

A big part of design and development process is automated using business rules, especially developing transactions and applications logic coding. Data structure is created using traditional DBMS; however, some of the restrictions and constraints can be described by business rules, connecting them with metadata about logical database. This gives to a programmer a possibility to enter needed changes without changing data structure.

Instead of rewriting transactions or recoding application code, appropriate business rules can be implemented and automatically executed. Programmer needs only once to relate business rules with data, that are impacted by the rule and this rule will be always invoked when appropriate data are changed. Business rules are invoked from database triggers; therefore there is no need to write additional code for this task.

Business rules repository acts as a common point of communication for both users and IT professionals. The advantages of developing systems using business rules are as follows:

- Rules are represented in a format that is understandable by users. Users can help enter and manipulate the rules.
- Rules are represented in a format that can be used by IT professionals as a system design document.
- Entered business rules can be immediately verified in the system.

A key difference between the traditional communication process and the business rules-based communication process is that the information in the repository can be understood and used by users. The only potential communication breakdown in this environment is during entering business rules into the repository. Since it is possible verify entered rules almost immediately, users can quickly interact with the system and evaluate whether or not the business rules have been accurately represented.

In the future work business rules editing interface will be improved in order to have business rules debugging features and business rules engine will be expanded with clear and thorough messages shown to user when business rule is violated.

References

[1] **T. Beale**. OCL 2.0 *Review*. 2003, *http://www.deepthought.com.au/it/ocl_review.html*.

[2] Blaze Advisor. *Ruling a self-service world*, 2001, *http://www.qualitywriter.com/Samples/advisor_selfservice.pdf*.

[3] Business Rules Group. *Response to MDC/Microsoft Business Rules Metamodel*. 1999, *http://www.businessrulesgroup.org/brg-mdc/BRG-MDC.pdf* .

[4] **R. Butleris, K. Kapocius.** The Business Rules Repository for Information Systems Design. 6*th East-European Conference ADBIS*'2002", *Research Communications, Bratislava: STU*, 2002, *Vol*.2, 64 –77.

[5] **R. Butleris, L. Motiejūnas**. A Framework for Business Rules Storing and Activating. 7*th International Baltic Conference on Databases&Information Systems*'2006, 252 – 263.

[6] **R. Butleris, L. Motiejūnas.** Metadata for business rules integration with database schema. *IADIS Virtual Multi Conference on Computer Science and Information Systems* (*MCCSIS* 2005), 263-270.

[7] **R. Butleris, L. Motiejūnas.** Veiklos taisyklių manipuliavimo mechanizmų analizė. *Informacinės technologijos* 2003, XIV-82 – XIV-90.

[8] **E. Gottesdiener.** Business Rules. Show Power, Promise. *Application Development Trends, Vol*.4, *No*.3. 1997, *http://www.ebgconsulting.com/Pubs/Articles/BusinessRulesShowPowerPromise-Gottesdiener.pdf*.

[9] Guide Business Rules Project, *Final Report*, 95/11. *http://www.businessrulesgroup.org/brgactv.htm*.

[10] **P. Kardasis, P. Loucopoulos**. Expressing and organising business rules. *Information and Software Technology*, 2004, *Vol*.4, *No*.11, 701-718.

[11] **C. Mariano C. Bornhövd, A. P. Buchmann**. Moving Active Functionality from Centralized to Open Distributed Heterogeneous Environments. *Lecture Notes in Computer Science*, 2001, *Vol*. 2172, 195-211.

[12] **J. Nicola, M. Mayfield, M. Abney.** Putting business rules into business objects. 2007, *http://www.streamlinedmodeling.com/papers/business_rules.pdf*.

[13] **A. Perkins.** Business Rules Are Meta Data. *Business Rules Journal*, 2002, *Vol*., *No*.1, *http://www.BRCommunity.com/a2002/b097.html*.

[14] **D. Plotkin.** Business Rules Everywhere. *Intelligent Enterprise Magazine*, 1999, 2 – 4, *http://www.iemagazine.com*.

[15] **R.G. Ross.** The business Rule Book (2nd ed.). *Business Rule Solutions*, 1997.

[16] **R.G. Ross.** Principles of the Business Rule Approach. 2003.

[17] **G. Wagner, S. Tabet, H. Boley.** MOF-RuleML: The Abstract Syntax of RuleML as a MOF Model. OMG's INTEGRATE 2003. *Boston, Massachusettss* 2003, *http://omg.org/docs/br/03-10-02.pdf*.

[18] **J.B. Warmer, K. Anneke.** The object constraint language: precise modelling with UML, 2000.

[19] **K.D. Wilson.** Business Rules, Platforms, and Inferencing. *Business Rules Journal*, 2003, *Vol*. 4, *No*.10. *http://www.BRCommunity.com/a2003/b169.html*.

Received July 2007.

DOI: 10.5755/j01.itc.36.3.11883