Managing Knowledge as Business Rules

Anca ANDREESCU, Marinela MIRCEA
Academy of Economic Studies, Bucharest, Romania,
anca.andreescu@ase.ro, mmircea@ase.ro

In today's business environment, it is a certainty that will manage to survive especially those organizations which are striving to adapt quickly and with low costs to the new demands of market competition. Knowledge represented by internal business rules of an organization can help crystallize their orientation in order to ensure a competitive advantage in the market. In this context and in a relatively short time, a new trend in software development has arisen, extending current methods and putting a strong emphasis on business rules. This article outlines the importance of managing business rules in an organized manner using dedicated software products and furthermore presents a general prototype for a business rules repository.
Keywords: *Business Rules, Management, Knowledge, Rule Engine, Repository Prototype*

1 Introduction

As interest for explicit manipulation of business rules grows, researches in this area seem to focus on a common goal: to identify ways that provide support for automatic propagation of changes from business environment to information systems, and further, to software applications. Thus, it aims to fill the gap between the business level and the information system level, because aligning information system with business functional requirements is a fundamental problem of all organizations. In the same time, it is essential to trace the business rules in all stages of the software life cycle, which can't be accomplished without addressing at least the following topics [1]:

- *Business rules identification*: How are business rules identified, starting from the business objectives and from the business stakeholders? How are business rules extracted from the legacy source code?
- *Business rules specification*: How to specify business rules, so that they are understood by all those involved in the development process?
- *Business rules implementation*: What kind of technology should be used to implement business rules? Where to implement business rules within an application, in order to minimize the effort required modifying the rules?
- *Business rules management*: Where to store business rules and their contents? How to manage business rules changes and versions?

The process of identifying business rules can be difficult especially where these rules do not benefit from an explicit representation. Depending on the information they contain, business rules may be based on explicit or tacit (implicit) knowledge. Explicit knowledge can be easily identified, formalized and expressed in the form of principles, terms, policies, formulas, etc. On the other hand, tacit knowledge is not as visible and easily expressed [2]. It has a highly personal and subjective nature and may be based on experience, ideals, emotions or intuition. The importance of tacit knowledge resides in its ability to clarify the context of business rules for which there is a not well-defined specification. To understand why a certain rule exists, and what its motivation is, one must first understand its information basis. Otherwise it is possible to lose the meaning for which the rule was created or no longer know if it still reflects reality [3].

In this respect, Scott Amber proposed in [4] a way to capture the information behind a rule by using a template containing the following fields: name, identification code, description, sample, source, document, publication date, related rules, and history of changes.

Once identified, business rules must be specified in an appropriate manner. Thus, a new challenge appears: that the method chosen for rules specification to be understood by all

persons involved in software development. While business people are not so often familiar with specification languages that require a higher level of formalization, developers require that business rules statements to be unambiguous in order to allow an easy transition towards source code. This contradiction leads to the following conclusion: to be understood by all people that are using them, business rules must be specified at different levels of formalization. Figure 1 depicts four levels of formalization at which rules can be described, so they allow the evolution from one specification method to another [5]. The main characteristics of these specification methods are also presented.

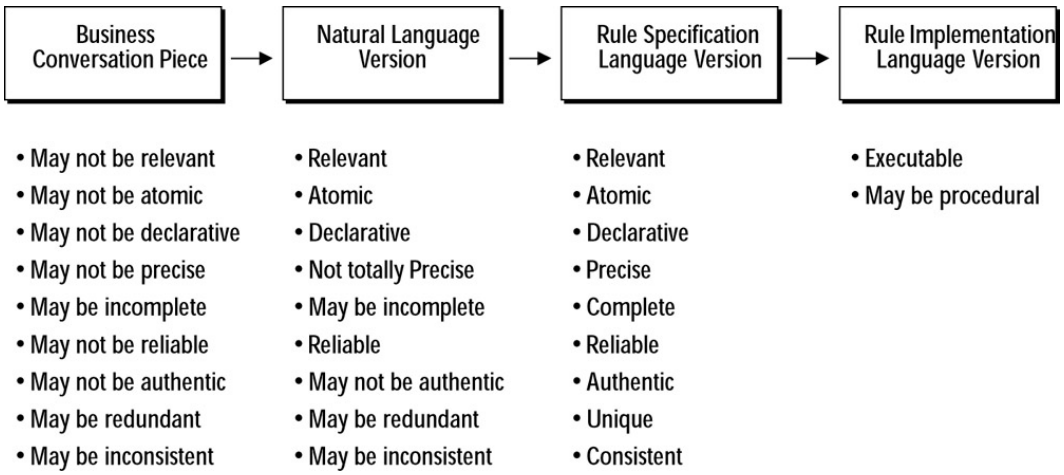

Fig. 1. Types of rule expressions [5]

So, in the context of different rules expressions and versions, a consistent business rules management is essential for the success of a rule-based development process.

2 Business Rules Management Systems - characteristics and architectures

Recent years have been marked by an increased interest for a new type of software products, called Business Rules Management Systems (BRMS). These systems externalize business rules and provide facilities for a centralized business rules management. They also offer solutions for compelling problems facing any business: business rules changes in response to increasingly rapid pace of change and the short time required for the implementation of there changes in the software system.

BRMS products are related both at conceptual and commercial level with expert systems that have emerged in the 1980s. Expert systems (also known as knowledge-based systems) represent the oldest and perhaps the best documented technology in artificial intelligence. These are systems that can offer suggestions or take decisions in a well defined area of expertise. There are two types of such systems: a) systems that make decisions and which mostly control processes such as systems for financial transactions and b) systems acting as decision support systems, which are not designed to take autonomous decisions. But the most important feature of expert systems is given by their architecture: the knowledge related to problem domain (represented, usually in the form of rules) is stored separately from the code that uses knowledge to solve a specific problem.

Expert systems are therefore based on rules. Rules must characterize, in a very complete and accurate manner, the universe of discourse and the decision-making context. The rules can be derived using data analysis techniques or various methods based on statistics. Also, rules can be arbitrary sentences imposed by prejudices and thinking of the decision-maker. Whatever are their sources and however rules are built, it was observed that declarative rules are sufficiently flexible for the proper representation of acquired knowledge, being in the same time easy to handle and implement [6].

In an expert system, rules must be properly

specified in order to capture the action that must be executed when an event is triggering it. A rule may have different forms depending on the implementation method or type of rule. The classical production rules format is the most used: "If A then B", where A is called antecedent clause and B is called consequent. In rules specification, the consequent usually takes the form of an action or conclusions. A production rule can have multiple interpretations, such as: when a condition is satisfied, then a particular action is executed; if a particular statement is true, then another can be inferred; if a particular syntactic structure is present, then another can be grammatically generated. In general, A and B are complex statements, formed from simple constructions using connectivity operators as disjunction, conjunction or negation. A substantial knowledge base can be created using a general specification method, called ECAA (Event-Condition-Action-Alternative-Action).

Table 1. Comparison between Expert Systems and Business Rules Management Systems

Expert Systems	**Business Rules Management Systems**
Are intended for independently solve one complex problem.	Solve a large number of relatively simple problems, using the form of service calls.
Rules generally represent expertise in a narrow field, understandable to few people.	Business rules capture general knowledge of the business area. They have a wide audience and are intended to be understood and validated by all the people that have an interest within the organization.
When arriving at a conclusion, the problem can be considered solved.	The rules are continuously applied, most organizations aiming for continuous operation of their applications, so that we can not say that such systems reach a specific end point.
Beside rules, they can process expert knowledge in order to provide answers, advices and recommendations.	Are appropriate for making decisions based on a large number of relatively simple rules.

General principles of expert systems mentioned above are also applicable to BRMS. Even if the two classes of products are related at conceptual level and even technologically, there is a series of arguments showing why they have evolved in different directions (table 1).

Unlike the first generation of expert systems, which mixed together facts, data procedures and rules in knowledge base, a modern BRMS keeps, in most cases, a clear demarcation between rules and business data. Knowledge stored in a BRMS is referenced as *rules base* or *knowledge base*, and the mechanisms that apply knowledge over data are called *rule engine*s or *inference engine*s.

According to Jan Graham [7], Business Rules Management Systems have four essential components illustrated in figure 2.

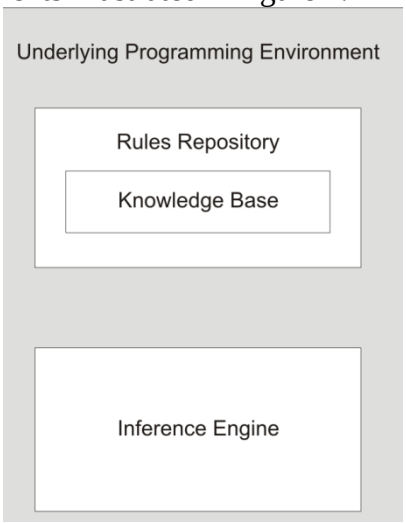

Fig. 2. Simplified architecture of a BRMS

The first element in the above figure consists

of the environment underlying the system and within which the system runs. It can contain programming languages and their development environments, text editors, compilers, processors, data structures, etc. The second element is the very structure of knowledge, including methods of representation and access, as well as techniques for applying knowledge in a rational way in order to solve problems. The third element is the inference engine, which establishes a link between the rules, for obtaining accurate and valid conclusions. In most cases, inference engines operate in a non-procedural way, but some BRMS allow rapid implementation of rule sets using procedural execution methods. The last element of this architecture is the repository, where rules are stored and which allows the application of various operations on rules, such as manipulation, versioning management, sharing, etc.

In a broader architectural context, the capabilities of BRMS must be centered on the rule-based decision services that it offers to basic applications (figure 3). Expert systems were traditionally designed as closed systems, which independently resolved certain problems and did not allow integration with other programming environments. This is probably one of the reasons why expert systems didn't have the expected success in developing business applications. Unlike expert systems, BRMS products were designed to provide services that automate business applications within an organization.

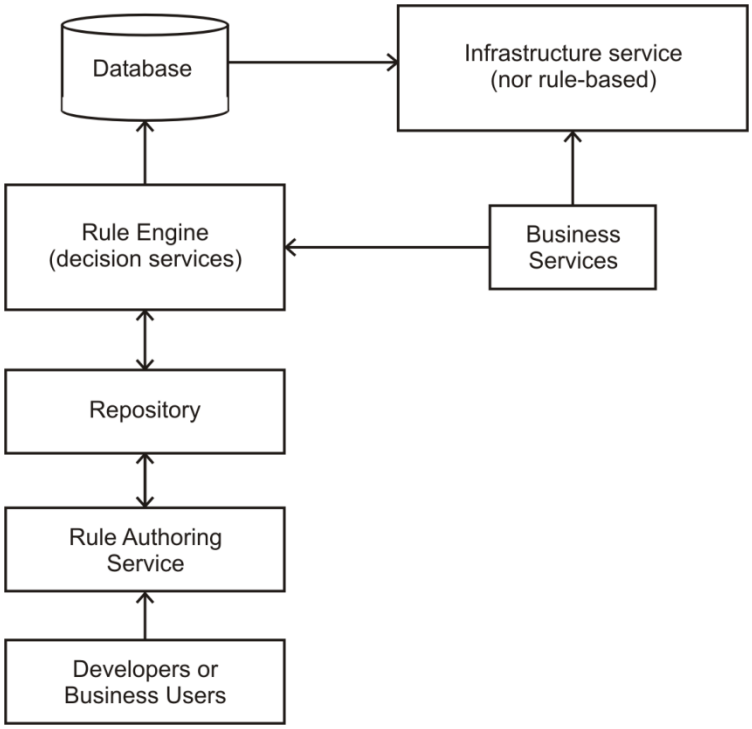

Fig. 3. Components of a BRMS (adapted after [7])

The separate placement of the rules repository must be observed (from which the rules will be imported to represent entries for decision services), but also the likely presence of one or more databases that, in a software system, are closely related to business rules. In the same time, business rules authoring services (responsible for business definition) are placed in a separate component. In order to facilitate the application maintenance, the knowledge base and inference engine are represented as separate entities. Since rules and policies are those that will change over time, it is not practical to rewrite the code associated with each rule engine each time a new rule appears.

Knowledge Base contains usually different types of knowledge about objects, procedures or causal relationships. Knowledge about objects may be stored as object models, XLM

schema, data models or semantic networks [7]. Procedural knowledge can be represented as rules, but they may be, for example, methods written in Java or macros in MS Excel. Meanwhile, some business procedures can be represented as rules.

As previous noted, knowledge about causal relationships is stored as rules of type "IF ... THEN". Programming languages such as C#, Java or COBOL implements such statements using control structures (if / then, while, until or case). Languages that implement business rules are typically declarative, hence non-procedural. This means that the order in which rules are written is not important. Rules apply to knowledge about entities and objects.

Among different ways of knowledge representation, rules, procedures and objects are, nowadays, the most commonly used by Business Rules Management Systems. In most BRMS products, rules are specified in the form of sentences, containing, usually, the words IF and THEN. However, Morgan proposed in [8] a different style, in order to reduce ambiguity, to establish explicit relationships between elements, to avoid obscure terminology, etc. His style is remarkably close to natural language.

The *template* concept, frequently used in software development, finds itself useful in rules-based technology as well. In this context, it allows the creation of sentences with blanks spaces that will to be filled in afterward. This reduces the time required to specify the rules and provides premises to enforce a standard style and unambiguous representation.

Most BRMS products allow or even require the placement rules that will to be executed together into a *set of rules*. The motivation resides in the need to associate rules governing a particular function of an application. For example, all rules that are related to discounts may be grouped in the set of rules "discount rules".

Rules *syntax checking* includes the possibility to check the syntactic correctness of a rule, in real time and as the rule is introduced into the system. In order to accomplish a clear link between object model and rules is useful to highlight keywords, variables and values using different colors.

It is obvious that an efficient business rules management process can't be achieved without the use of suitable instruments for this purpose. There are plenty of such instruments on the market, that provide facilities for business rules acquisition and management, each covering a specific area of rules life cycle and addressing to different categories of users. Thus, by analyzing the existent solutions, three main classes of software products based on business rules have been identified. These are described below.

Class A includes products for which business people represent the main audience. They are independent of a particular development environment and perceive business rules management from a business perspective. They provide specialized services for the rules acquisition, including identification of those business artifacts that allow placing business rules in a certain context. The offer of such products is relatively small, their main drawback being that they don't support rules implementation. Typical products for this class are RuleXpress (RuleArts) and RuleTrack BRS (Business Rule Solutions).

Class B includes products intended primarily for developers, especially analysts and software architects. Their role is to assist the development of rule-based applications, by providing facilities for rules acquisition, formalization, modeling and, especially, for implementation. In most cases, they don't offer support for non-technical users, such as high-level specification languages. Many products on the market belong to this class: InRule (InRule Technology), VisualRules (Innovations Software Technology), Usoft (Ness Technologies), Versata BRMS (Versata) etc. Even though, technically speaking, they represent only parts of a BRMS, rule engines correspond to a technology that can be used in software development independently of any other products. Several rule engines must be noted: open source rules engines such as OpenRules and Drools for Java and NxBRE or Drools. NET for the .NET platform, but

also the rule engine included in Microsoft technology, called Windows Workflow Foundation.

Class C has the widest audience and is open to all categories of persons involved in the development of knowledge-based applications. Being built on the paradigm of expert systems, they allow the creation of intelligent applications based on knowledge acquisition. They are versatile products that have powerful inference engine and provides a wide range of facilities for rules management, both at business level and at software system level. Blaze Advisor FICO (FICO) and ILOG Rules (IBM) are recognized as market leaders in this segment.

3 Evaluation criteria for BRMS products

Most companies that offer BRMS products have evolved from the status of rule engines vendors to that of software development solution providers for the business environment. These solutions were based on the definition of declarative business rules that run through proprietary rule engines. Besides these, there are also producers that have started from different approaches, such as mapping decision trees or graphs to executable code (for example, Visual Rules product). The rules stored in the repository can be considered as decision services that are often compatible with the latest trends in software architecture, such as SOA or Web services.

Aspects regarding the opportunity and the decision to implement a BRMS product and a complete business rules approach were discussed in detail in [9] and [10]. This article aims to recommend a set of evaluation criteria for BRMS products and, afterward, these criteria will be applied for the following products: FICO Blaze Advisor, Visual Rules and Windows Workflow Foundation (WWF). The criteria were divided into two categories, as follows:

a) General criteria:

- *Affiliation to a class of rule based products* in accordance with the three classes proposed above and which identify the product target audience.
- *Price*, especially for rule-based products, significantly reduces the range of potential clients.
- *Type of target organization*, because for medium or large organizations, the analysis of cost-effectiveness may lean in favor of implementing a BRMS, while the vast majority of small companies can not afford to purchase such a product.

b) Technical criteria:

- Allows *backward and/or mixed chaining strategies*, as there are systems intensively based on knowledge, for which the opportunity to implement rules using backward or mixed chaining is crucial.
- Uses a *rule engine based on Rete algorithm*, particularly useful for systems containing many interrelated rules.
- Uses a *high-level language for business rules implementation*, aiming the kind of systems that will or should allow business people to handle, modify or create business rules.
- Represents rules in *graphical form* as decision tables, score tables or decision trees for modeling complex rules.
- The *software platform* on which the BRMS product runs is important because it must be compatible with the base applications platform.
- Provides the opportunity of *testing rules before implementation*, being an important feature that relieves developers of testing rules by using other tools.

Table 2 presents the evaluation of three BRMS products, by reference to the above criteria.

Table 2. BRMS products evaluation

Evaluation Criteria	Visual Rules	Blaze Advisor	WWF
Class of rule based products	B	C	B
Purchase price	20.000- 70.000 €	from 200.000 €	free
Type of target organization	small and medium	large	any type
Allows backward/mixed chaining	No	Yes	No
Uses a Rete-based rule engine	No	Yes	No
High level language for rules	No	Yes	No
Graphical representation of rules	Yes	Yes	No
Software platform	Java	Java and .NET	.NET
Facilities for rules testing	Yes	Yes	No

Finally, in terms of usability, the analyzed products are not very suitable in environments where business people must be involved directly in the creation and maintenance of business logic. An exception is the situation when the development environment allows the creation of very intuitive custom interfaces, as is the case of the web interfaces provided by FICO Blaze Advisor. Visual Rules is not suitable to integrate rules into an existing application, but to build a new application with rule-based technology.

4 A business rules repository prototype

It is clear that, regardless of how the rules are introduced in the software development process, handling them effectively requires the support of a rules repository, in the same way that data management requires centralized databases storage.

A rules repository allows the storage, outsourcing and distribution of the rules that have been identified in the business modeling phase and in other phases of the development process. Generally speaking, a repository is a specialized type of database that allows the storage of complex objects, together with their description [11]. In software development, these objects can be models, components or specifications. Both the requirements and the conceptual structure of rules repositories have been the subject of several scientific papers or publications in this field, such as [12], [13] or [14].

An example of a logical data model for a rules repository, having a high degree of generality, was described by Tony Morgan in [8].The author addresses the problem of data repositories' utility and highlights their individual character when compared with other tools for managing rules. Morgan suggested the use of constructions based on XML as a solution to the multitude of formats, groups of rules or taxonomies.

In the same register, Marko Bajec's and Marjan Krisper's methodological research [15] has resulted in a system prototype for building a business model. This model includes specific business elements, such as: objectives, problems, activities, processes, organizational and functional units and provides support for the incorporation and management of business rules in this model. Yet again, rules are treated at the two levels mentioned above.

Generally speaking, by using rules repositories, it is intended to achieve the following benefits: ● a more effectively and quickly management of business requirements; ● better tracking and reuse of business rules; ● increasing rules security and integrity within the system.

The following paragraphs will present the main objectives and functionality of our proposal for a business rules repository prototype, which was named RuleManager. It was created as a useful tool for developers in defining and managing business rules of an organization, being independent of any other development tool. The prototype application must be used within a development process with capabilities for explicit manipulation of business rules, having two types of users: business analyst and software architect, for which a general use case diagram is described in figure 4. Such a process, extending the Unified Software Development Process [16] with additional activities proper for

business rules manipulation, can be found in [10].

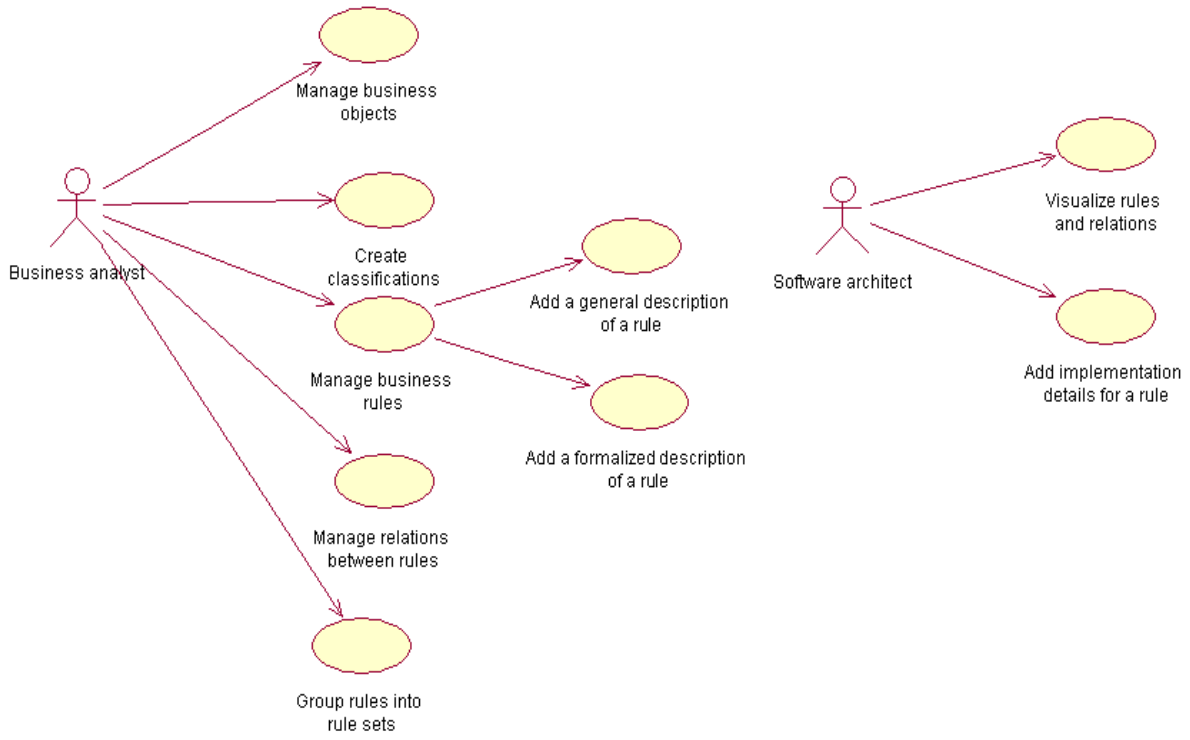

Fig. 4. General use case diagram

According to this prototype, the *business analyst* plays the most important role in the application, acting as an interface between business people (system's beneficiaries) and developers. His main tasks are related to informal and structured specification of business rules, defining the relationship between rules and grouping rules into sets of rules. On the other hand, the *software architect* uses the application after the business analyst has entered all the details of a rule. On this basis, the software architect may take design-level decision. Finally, he must introduce in the application information regarding the implementation of rules.

As a rules repository, RuleManager has the following functions:

- To provide support for documenting business rules at three different levels of abstraction: *business level* (informal description), *software modeling level* (formalized description) and *implementation level* (implementation details).
- To provide rules traceability in a system.
- To enable modeling of different categories of rules, including complex rules such as decision tables.
- To establish relationships between rules.
- To group rules into set of rules according to functional or organizational criteria.
- To provide, through its interfaces, the possibility to manipulate objects from repository in a consistent manner.
- To provide the possibility of granting access rights to individual rules, groups of rules or rules that meet certain conditions.

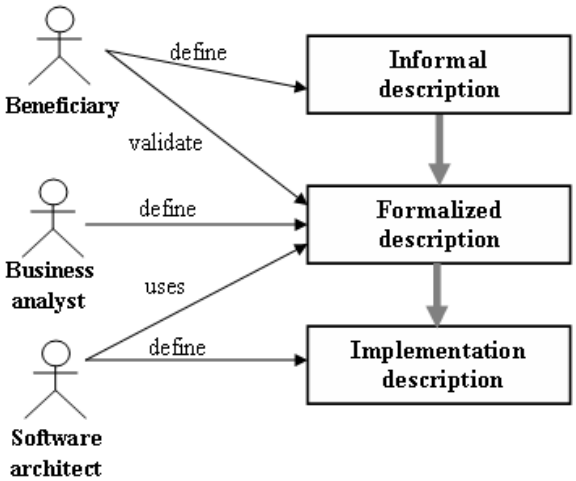

Fig. 5. Actors and their roles in the rules specification activity

In order to ensure that the prototype meets the functionality associated with these objectives, the biggest obstacle was to choose a proper method for their formalized description. As shown in figure 5, the formalized description represents a bridge between customer requirements and the actual system implementation.

According to the scenario in the above figure, the formalized description is defined by the business analyst, based on the beneficiary's informal description, is validated by the customer and is used by the software architect to serve as a basis in choosing an implementation solutions. Thus, this approach started from the idea that the formalized description should be understood by all these people involved in the development process. In order to specify rules, RuleManager structures a rules model as a tree structure, based on three main elements: *entities, rules and relationships*.

Entities are those which will form the basis of the formalized specification of the rules, while relationships capture the links between rules. Entities are, in fact, business objects found in the object model defined in the business modeling phase or in the system analysis phase. Each entity has an associated set of attributes, which are assigned to a specific type of basic data: numeric, string or date and time. Certainly, these entities are not isolated and they must be correlated to other entities in the model. The application does not model the dependencies between entities and assumes that they were represented as a class diagram or model of facts.

The prototype allows the definition of four categories of ***business rules***, as follows:

- *Enumeration rules* establish a set of possible values that an attribute of an entity may take.
- *Integrity rules* capture some constraints that are applied to an entity's attributes.
- *"If…then"* rules are equivalent to Event - Condition – Action (ECA) rules or production rules.
- *Decision table rules* are used to represent complex rules in a tabular form, and allow making associations between several interdependent conditions, on the one hand, and defining a set of actions that correspond to a combination of these conditions, on the other hand. The main advantage of decision tables lies in the intuitive way in which they represent knowledge, in order for this to be better understood and validated by humans.

For each rule, the user must specify some general information, a formalized description and implementation details. Regarding the general information, they are meant to uniquely identify a business rule within the system and also to provide a set of other useful details for implementation. The prototype refers to following general fields:

- **rule code**: must contain a unique identification code for a rule; it can be a meaningful name for the rule or a combination of characters and numbers to encode, for example, the type of rule, as in the following examples: BR12, ER05 or DTR23.
- **description**: refers to a brief characterization of a rule.
- **details**: contains the specification of a rule in an informal manner, namely in natural language. This information is provided by the beneficiaries and is a part of the system requirements.
- **source**: represents, from business perspective, the basis of a business rule, because rules characterize the business, and not the developed system. These sources can be, for example, organization's internal policies or legal regulations coming from outside the organization.
- **category**: allows the inclusion of a rule in a set of rules, for the reasons that were mentioned earlier on.
- **validity**: refers to the period during which the rule will be operational.
- **mode**: describes the dynamic character of a rule, and may take two values - static and dynamic. It can be very useful for decision makers to know the possibility of a rule to change frequently. Ideally, all rules should be implemented so that they can be changed without affecting the sys-

tem modules that implement business logic and, especially, without requiring a recompilation of the entire application. Rules-based development principles promote the very idea of implementing all rules in an external component, where rules can be managed by business people and executed by a rules engine. The reality, however, differs from this ideal scenario, and the characteristics of particular rule, of the system, and not least, the available resources of the beneficiary, may constitute impediments for using a rules engine.

- **status**: helps establish important moments in the life cycle of a rule. At the time when it was created, a rule state must be *proposed*. Once it was validated by beneficiary, together with the business analyst, it passes the *approved* state. An approved rule will be implemented until it becomes operational, moment set by specifying the range of validity. After adding the implementation details, the rule becomes *active*. Finally, there is the possibility that the rule to become *inactive* due the expiration of its validity interval, a decision taken within the organization or other reasons.

For the formalized description of a rule, the prototype provides editing features specific to each category of rules. For example, for editing "if...then" rules, it was taken into account the fact that the user must have, in terms of semantic, more flexibility in defining the conditions of a rule, but syntactically, he must use a predefined set of symbols . Beside this, "if…then" rules formalized definitions must comply with the following pattern:

If [( ] <condition>
[<condition > [ )]
[( ] AND /OR < condition >….] [ )]
then <action>
[<action >
< action > …].

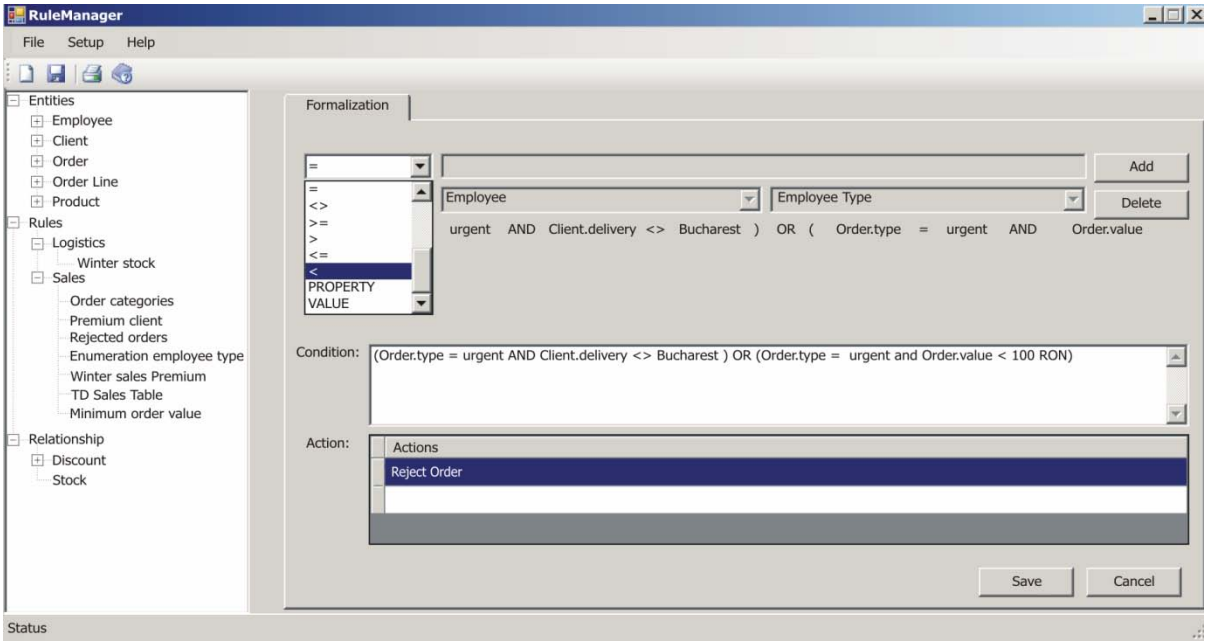

Fig. 6. The editor for the formalized description of an "if…then" rule

As shown in figure 6, the introduction of any element in the definition of such rules is subject to selection from a list, if these elements are logical operators (AND, OR), mathematical operators (+, -, *, /, =, <>, <=, >=, =) or round brackets. To refer to a property of an entity, the option PROPERTY must be selected and, in order to introduce values associated to a property, the option VALUE must be selected. Actually, Figure 6 illustrates how to edit the formalized description for the following rule: "If an order is urgent and the

shipping address is outside of Bucharest or if the order is urgent and order value is less than 100 RON, then the order will be rejected."

Also, note that the editor allows the insertion of such items anywhere within an already defined condition by selecting one or more elements that are placed before the ones that we want to add. Regarding the actions, the user is free to enter any specification. Therefore, the prototype allows easy rules editing, however, in a controlled manner.

The establishment of ***relationships between rules*** is an important feature of a rules repository, since, in a system, many rules are in close connection with other rules. The prototype implements one method of connecting rules, according to which between two rules may exists three types of relationships, as in the subsequent definitions:

Definition 1: *Rule R1* ***includes*** *rule R2, if for the establishment of the truth value of R1 if it is necessary for R2 to be previously executed.*

Definition 2**:** *Rule R2* ***extends*** *rule R1, if R2 re-evaluates, for a particular case, the conclusions established through the evaluation of R1.*

Definition 3**:** Rule *R2* ***excludes*** *rule R1, if R2 stipulates an exception from R1.*

It must be mentioned that, at the level of relationships, rules can be grouped into sets of rules other than those encountered in the rules definition. Thus, in case of the example in Figure 6, a new set of rules may be needed (named Discount), in order to contain the relationships between the rules governing discount policies.

5 Conclusions

Business rules encompass knowledge about specific politics, procedures, definitions or legal regulations that every organization must comply with. If accurately managed, they may lead to relevant competitive advantages within the business environment. This article analyzed the fundamental principles that govern rule-based development and the particularities of Business Rules Management Systems, by providing a set of evaluation criteria for this type of software products. In order to illustrate the need and usefulness of a mechanism that allows business rules management during the software system development cycle, a solution for a business rules repository prototype was proposed. In brief, this prototype is independent by any development tool and addresses those rules that will be directly implemented in the system. Future research will include the definition and addition of supplementary functionalities to the repository prototype, such as: ● handling collections of business objects;● code generation for rules programming languages or specification languages, like RuleML;● management of rules versions;● import business object model, as a database structure or class model.

References

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

[2] I. Nonaka, H. Takeuchi, *The Knowledge-Creating Company*, Oxford University Press, Oxford, 1995.

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

[4] S. Ambler, *Object-Oriented Business Rules*, Software Development Magazine, June 2000.

[5] B. Von Halle, *Business Rules Applied*, John Wiley & Sons, Inc., 2002

[6] R. Petruşel, *Sisteme suport de decizie în domeniul fluxurilor de trezorerie aplicate la întreprinderile mici şi mijlocii din România*, PhD thesis, "Babes-Bolyai" University, Cluj-Napoca, 2007.

[7] I.Graham, *Business Rules Management and Service Oriented Architecture: A Pattern Language*, Wiley, 2007.

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

[9] A. Andreescu, A. Uta, *Methodological Approaches Based on Business Rules, Informatica Economică Journal* [Online], no. 3(47). Available:

http://revistaie.ase.ro/content/47/04Andreescu,%20Uta.pdf

[10] A. Andreescu, *A General Software Development Process Suitable for Explicit Manipulation of Business Rules*, The proceedings of the "CompSysTech '09" International Conference, Rousse, Bulgaria, pp. IIIA.9-1, June 2009.

[11] P. A. Bernstein, U. Dayal. *An Overview of Repository Technology*, in proceeding of VLDB'94, pp. 705-713, Santiago, Chile, 1994.

[12] N. Haggerty, J. Wall, G. Van Etten, *Defining The Requirements for a Business Rule Repository*, The Data Administration Newsletter, no.16, 2001. Available: http://www.tdan.com/view-articles/4924

[13] R. Butleris, K. Kapoèius, *The Business Rules Repository for Information Systems Design*, ADBIS Research Communications, 6th East European Conference ADBIS, Bratislava, 2002

[14] L. Čeponis, O. Vasilecas, *Logical View of Architecture of Business Rules Repository*, Information Sciences, no.36, 2006. Available: http: //www.ceeol.com

[15] M. Bajec, M. Krisper, *A methodology and tool support for managing business rules in organizations*, Journal Information Systems**,** Vol. 30, no. 6, Elsevier Science Oxford, September 2005.

[16] G. Booch, J. Raumbagh, I. Jacobson, *The Unified Software Development Process*, Addison-Wesley, 1999.

Anca Ioana ANDREESCU is university lecturer in Economic Informatics 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, requirements engineering and software development methodologies.

Marinela MIRCEA is assistant professor in Economic Informatics Department, Academy of Economic Studies of Bucharest. She published over 30 articles in journals and magazines in computer science, informatics and business management fields, over 15 papers presented at national and international conferences, symposiums and workshops and she was member over ten research projects. She is the author of one book (currently in issue) and she is coauthor of three books. In February 2009, she finished the doctoral stage, and her PhD thesis has the title *Business management in digital economy*. Her interest domains are: programming, information system and project management.