Conceptual Modeling of Business Rules and Processes with the XTT Method*

Grzegorz J. Nalepa[1] **and Maria Antonina Mach**[2]

[1] Institute of Automatics,
AGH University of Science and Technology,
Al. Mickiewicza 30, 30-059 Kraków, Poland
<*gjn@agh.edu.pl*>
[2] Institute of Business Informatics,
University of Economics,
Komandorska 118/120, 53-345 Wrocław, Poland
<*maria.mach@ae.wroc.pl*>

Abstract *The paper deals with the problem of modeling business rules (BR) in the business process management (BPM). The paper presents the challenges of BR modeling, and shows some disadvantages of existing, commonly used methods, such as the BPMN. As a solution, an approach based on the XTT/ARD expert system design and implementation method is outlined and discussed. The methods improve the design and modeling of BR, leading to a more complete, and transparent representation of the rule base.*

1. Knowledge in Business Process Management Let us begin with a definition of *Business Process Management* (BPM). In [13] we find the following one: "supporting business processes using methods, techniques and software to design, enact, control and analyze operational processes involving humans, organizations, applications, documents and other sources of information". This definition is of extremely high importance, because one can immediately see the main issues or problems connected with the BPM. These are the questions of incorporating human knowledge into BPM systems, knowledge representation, and mining knowledge from BPM systems. Therefore, BPM is strictly linked with such areas of Artificial Intelligence (AI) as Knowledge Management (KM), Knowledge Engineering (KE) and Business Intelligence (BI).

*The paper is supported by the *HEKATE* Project funded from 2007–2009 resources for science as a research project.

Knowledge is an essential factor in practical BPM. Knowledge related issues include: acquisition, representation, evaluation, and processing. Knowledge representation methods need proper syntax, visual representation, and formal foundations. These issues have been extensively studied in the field of Knowledge Engineering. Before applying any of the knowledge representation techniques, knowledge on processes has to be gathered and acquired both from existing systems and people. While the first task is not very complicated, the second one is not trivial.

In AI, *rules* are probably the most popular choice for building knowledge-based systems (KBS), that is the rule-based expert systems [3, 5]. Rule-based systems (RBS) are used extensively in practical applications, especially in domains such as automatic control, decision support, and system diagnosis. They constitute today one of the most important classes of KBS. Practical construction of a rule-based knowledge base, also referred to as the *rulebase* involves number of important steps. These include rule attribute specification, rules design, evaluation, and a practical implementation.

Recently, a new approach to practical knowledge representation based on rules, has been gaining popularity. This is the so-called *Business Rules Approach* (BR). As stated in [2], "a business rule is a statement that defines or constraints some aspect of the business. It is intended to assert business structure or to control or influence the behavior of the business". Further information on the BR approach is given in Section 3.

The focus of the paper is on the design and modeling of BR in the BPM (Sect. 2). It includes the discussion of the most important issues concerning practical design of business rules (Sect. 3), found in the BPM. In the paper a new design method is described, using an example business rulebase described in Sect. 4. This example has been originally designed using the BPMN (Business Process Modeling Notation) [10], and accompanied with business rules. The method presented in this paper, is centered around the XTT approach, presented in Sect. 5. The Sect. 6 discusses, how applications of these methods could improve aspects of BPM. The paper ends with concluding remarks in Sect. 7.

2. Business Process Modeling The main features of *business process modeling* are: *descriptive* what happens during a business process, in what way the process has been performed, what improvements have to be made; *prescriptive* allows for a definition of a business process and how a process should be performed, it lays down rules, guidelines and behavior patterns; *explanatory* links processes with the requirements explains the rationale of business processes.

These aims lead to the formulation of the requirements that a business process model has to fulfill. First of all, a model has to provide a holistic approach dealing with organizational and technical issues [4]. Next, BP models should have a strong formal foundation. It is so because formal models are unambiguous, and increase the potential for analysis [13].

There are several techniques for BP model specifications. Some are based on Petri nets [13], some make use of the UML notation. It must be nevertheless pointed out that UML, although widely used and adapted, is not expressly designed to map to business execution languages. The two formalized approaches to business process modeling, that are worthy mentioning are the *Business Rules Project* [2] and the *Business Process Modeling Notation* [10].

3. Business Rules Concepts and Tools Business Rules (BR) approach [12, 14] is based on concepts borrowed from knowledge engineering and rule-based systems. It is becoming an important approach in business application development, especially on the Java platform.

A classic description of the main principles of the approach is given in [12]. According to it, rules should be: written and made explicit, expressed in plain language, motivated by identifiable and important business factors, single sourced, specified directly by people who have relevant knowledge, managed, and built on facts, and facts should build on concepts as represented by terms. Rules should also exist independent of procedures and workflows. There are number of *rule types* identified in the BR approach, e.g. reactive, transformation, derivation rules.

Business rules design uses some established *visual representations*. Depending on the design approach these are some classic tools such as simple propositional decision tables, or some high-level conceptual tools such as URML [6]. There are attempts to officially define main aspects of the approach. A good example is the *Semantics of Business Vocabulary a Business Rules Specification*, see [11].

From the point of view of formal knowledge engineering, some major issues can be pointed out. They are related to: a) logical foundations, b) visual representation, and c) formal analysis and verification of BR systems.

The *first problem* concerns the *logical foundations* of BR systems. From a point of view of classical KE, a rule-based expert system consists of a knowledge base and an inference engine. The KE process aims at designing and evaluating the knowledge base, and implementing a proper inference engine. The process of building the knowledge base involves the selection of a knowledge representation method, knowledge acquisition, and possibly low-level knowledge encoding. In order to create an inference engine a reasoning technique must be selected, and the engine has to be programmed. In the formal analysis of RBS [5] these important aspects of the design and implementation are identified and analyzed.

Unfortunately it can be observed, that common approaches to BR tend to mix these formal aspects. The concept of “business rules types” is both misleading and imprecise. A proper formal analysis of BR should provide a more adequate classification of BR.

The *second problem* is related to the *visual representation* used in the design of BR systems. Visual representations used, have scalability problems (it’s easy to draw diagrams of several rules, but it becomes very difficult to cope with tens of rules). Lack of well-defined formal foundations of these representations leads to problems with an automatic transformation of the visual model to a logical one.

The *third problem* concerns the *formal analysis end verification* of BR systems. As the number of rules exceeds even relatively low quantities, it is hard to keep the rule-base consistent, complete, and correct. These problems are related to knowledge-base verification, and validation. The selection of appropriate software tools and programming languages is non-trivial either.

These issues are very rarely considered in the BR design. It seems that analysis (where issues such as verification, validation, and evaluation are even not properly separated) is simply considered testing. So the analysis of the *knowledge base* is implicitly substituted by testing of the *implementation*. However, in the KE approach, a proper analysis of the knowledge base minimizes the need for testing.

4. Business Rulebase Design Example Let us consider a classic illustrative BR example, presented on the Business Rules Forum in 2005 [1]. The example concerns the UServ Financial Services Company, which provides a full service portfolio of financial products, including: Insurance, and Banking. UServ plays a balancing act between rewarding their best clients and managing the risk inherent in providing on-going service to clients whose portfolios are profitable, but violate the eligibility rules of individual products.

UServ’s business rules are an essential component for managing this risk. The business rules address eligibility, pricing and cancellation policies at both the individual product and portfolio level. The case study [1] focuses on UServ’s vehicle insurance products, but differentiates the basic business rules from those that apply to preferred and elite clients. In the BR rulebase three groups of rules are identified: Client Segmentation Business Rules, Eligibility Business Rules, and Pricing Business Rules.

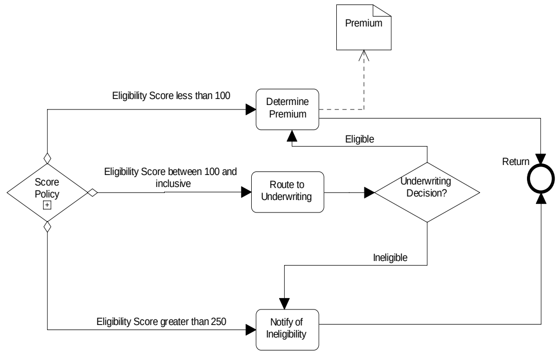

Figure 1. The BPM of Policy Processing.

The practical design of the system presented in [1] has been carried out using the *Business Process Modeling Notation* [10]). This is the main notation used in the BPM (see also Section 2). The main subprocess of the vehicle insurance policy processing can be observed in Fig. 1. In this process the scoring of the policy is delegated to a subprocess shown in Fig. 2.

This model is accompanied with business rules specific to the activities. Every activity in the diagram includes number of rules. So in this case the BR approach is being integrated on a low-level with use of rules as the way of expressing activities in given contexts. The main structure of the process is modeled using the BPMN.

In the following section an alternative way of integrating is proposed. It consists in using an advanced rule design and implementation process. The process is centered around the XTT visual and logical design method for RBS. XTT has a very rich semantics, which allows for designing number of systems, including business-related. The process is transparent and coherent, offering design methods that allow for both high-level conceptual design, as well as low-level rule implementation.

5. The XTT Approach The main idea behind *XTT* [8] knowledge representation and design method aims at combining some of the existing approaches, namely decision trees and decision tables, by building a special hierarchy of Object-Attribute-Tables [5]. It allows for a hierarchical visual representation of the OAT tables linked into tree-like structure, according to the control specification provided. XTT as a design and knowledge representation method offers transparent, high density knowledge representation as well as a formally defined logical, Prolog-based interpretation, while preserving flexibility with respect to knowledge manipulation. Formal definition of the XTT rule is:

$$\begin{aligned}\texttt{rule(i)}:\ & \texttt{context} = \psi \ \ \texttt{and} \\ & [A_1 \in t_1] \wedge [A_2 \in t_2] \wedge \ldots \wedge [A_n \in t_n] \\ & \longrightarrow \\ & \texttt{retract}(B_1 = b_1, B_2 = b_2, \ldots, B_b = b_b), \\ & \texttt{assert}(C_1 = c_1, C_2 = c_2 \ldots, C_c = c_c), \\ & \texttt{do}(H_1 = h_1, H_2 = h_2, \ldots, H_h = h_h), \\ & \texttt{next(j)}, \texttt{else(k)},\end{aligned}$$

where ψ defines the specific conditions under which the rule is valid, $[A_1 \in t_1] \wedge \ldots \wedge [A_n \in t_n]$ is the regular precondition formula, $B_1 = b_1, B_2 = b_2, \ldots, B_b = b_b$ is the specification of the facts to be retracted from the knowledge base, $C_1 = c_1, C_2 = c_2, \ldots, C_c = c_c$ is the specification of the facts to be asserted, $H_1 = h_1, \ldots, H_h = h_h$ is the specification of conclusions forming a direct output of the rule, *next(j)* is the control specification. On the Prolog level, rules are mapped to a *Horn clause*-based representation in the First Order Predicate Logic.

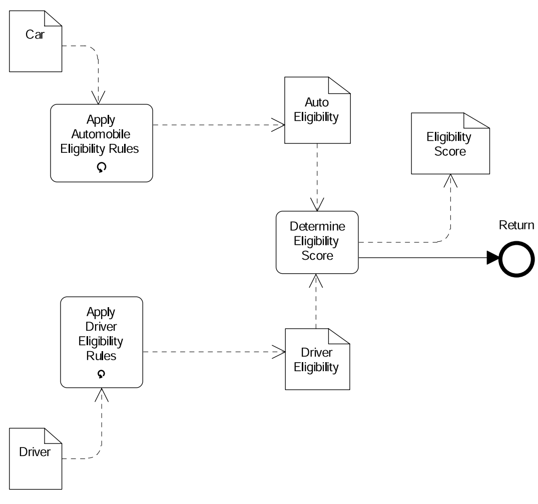

Figure 2. The BPM of Policy Scoring.

The *conceptual design* of the RBS aims at modelling the most important features of the system, i.e. attributes and functional dependencies among them. ARD stands for *Attribute-Relationship Diagrams* [7, 5]. It allows for specification of functional dependencies of system attributes using a visual representation. An ARD *diagram* is a conceptual system model at a certain abstract level. It is composed of one or several ARD *tables*. If there are more than one ARD table, a partial order relation among the tables is represented with *arcs*. The ARD model is also a hierarchical model. The most abstract level 0 diagram shows the functional dependency of *input* and *output* system attributes. Lower level diagrams are less abstract, i.e. they are close to full system specification. They contain also some intermediate conceptual variables and attributes.

Using XTT, an *integrated design process*, covering the following phases has been introduced:

- *Conceptual modeling*, when system attributes and their functional relationships are identified with use of ARD.
- *Logical design with on-line verification*, during which system structure is represented as XTT hierarchy, which can be instantly analyzed on-line, using Prolog.
- *Physical design*, in which a preliminary Prolog-based *implementation* is carried out.

A prototype CASE tools for the XTT method have been developed. They support XTT-based visual design methodology, with an integrated, incremental design and implementation process, providing the possibility of the on-line, incremental, verification of formal properties. They offer a direct translation to a formal, Prolog-based representation equivalent to the XTT knowledge base [9], with an *on-line* knowledge analysis, including formal verification, and refinement of the knowledge base. Let us now show, how the UServ example can be designed using this approach.

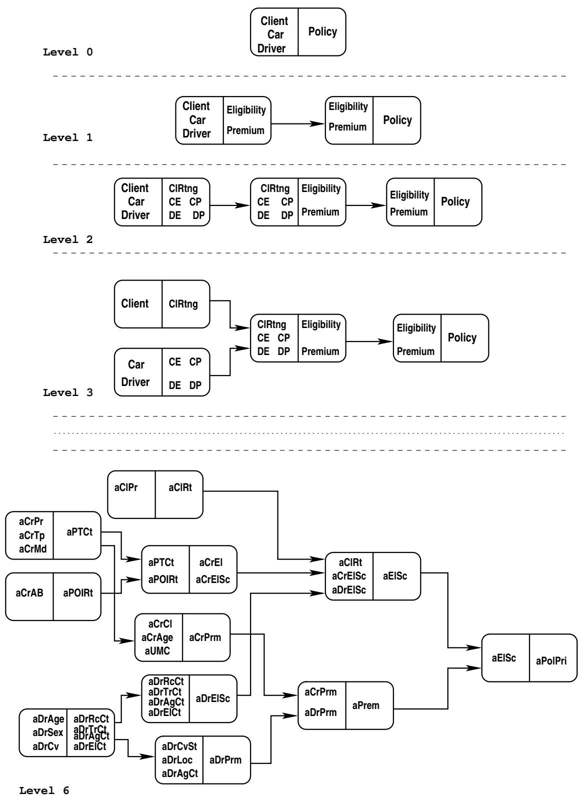

Figure 3. ARD/XTT Conceptual Design.

6. Visual Business Rules Design with XTT

The conceptual design with ARD is based on the idea of the full input/output specification, of the object attributes involved in the business process; in this case these would be: policy (for the car), client (that buys the policy), driver (of the car), the car itself. In the this design phase a complete specification of system attributes has been formulated. The design diagram can be observed in Fig. 3. ARD provides a hierarchical model. At the top-most *Level 0*, the most basic relation between some general input and output attributes is visualized. In this case, the level 0 diagram states, that „*policy pricing* depends on some features of the *client*, *driver*, and the *car*." At every subsequent level, this relation is becoming more specific, with the introduction of the physical attributes, and detailed relations between them specified. At the last, bottom level, the diagram gives full specification of relations between physical system attributes. In the figure, some of the middle levels are omitted. For more details on syntax and semantics of ARD see [7].

During the ARD design, two classes of attributes are used: *conceptual* and *physical*. The so-called *conceptual* attributes, or the *generalized* attributes, get specified during the conceptual design into the physical ones, present in the final rules. The *physical* attribute specification in the XTT/ARD contains: attribute names e.g. `aPotentialOccupantInjuryRating`, abbreviated attribute names, more suitable for compact Prolog implementation, e.g. `aPOIRt`, attribute types, e.g. `symbolic`, and specification of attribute value domains, e.g. `[Low,Moderate,High]`. The last level of the ARD is a *table scheme specification* for the XTT logical design method. During this phase, rules are built using specific attribute values. Using the results of the conceptual analysis, the actual design of the rule base is put forward, using the XTT representation method [9].

The XTT logical design involves the use of using the bottom level of ARD design, as a base for tables containing rules. The lowest ARD level includes all of the physical attributes present in rules. In a general case, the inference process is non-monotonic, since XTT allows for dynamic modification of the knowledge base, using Prolog-like *assert/retract* statements. The XTT structure corresponds to the structure of the decision process involved in the business process.

The XTT representation is automatically transformed into a Prolog-based representation [9]. Pure Prolog clauses are not used here, since Prolog uses backward chaining only. A meta-interpreter for forward chaining rules is provided. The Prolog-encoded rule-base is an executable prototype of the system.

The automatic transformation of XTT to Prolog, which can be done at any stage of the logical design, allows for an *on-line* evaluation of the rule base. In the prototype design tool number of Prolog-based verification plugins are provided. They verify some important formal properties of the system, such as redundancy, completeness, or determinism. What is important in this case, is the possibility of *formal* analysis, *during* the design. The details of the evaluation procedure are out of scope of this paper (see [9]).

Compared to the BPMN representation and design, the XTT/ARD model is complete and transparent. It offers a hierarchical representation of the whole knowledge base. While the ARD could be compared to BPMN design, the XTT allows for direct rule modeling, and the Prolog transformation gives an executable prototype. Both BPMN and XTT approaches have strong formal foundations. Yet some features of the latter solution make it a more advanced approach. These include automatic translation of the design specification into a low-level code, a well-defined system semantics and the possibility of on-line verification. The XTT design process provides a single coherent design semantics which does not mixes declarative rule-base representation, with procedural process specification.

7. Concluding Remarks The problem of designing and modeling the business rules is the core question in the BPM. Many solutions to this problem have been proposed and used so far, each them having both advantages and disadvantages. In the paper, the role of knowledge in the BPM, which leads to the question of proper knowledge representation in the form of business rules, is pointed out.

The Business Rules approach was outlined, and the main disadvantages of common approaches to BR design and modeling were pointed out, as well as the most important problems to be solved. As a solution to these problems, the XTT approach was proposed. It provides a coherent representation of rules, yielding a single formal representation of the main aspects of an enterprise's structure and operations. The possibility of on-line evaluation of the rule base, provided by the XTT approach assures correctness of the system being built.

References

[1] BRForum. Userv product derby case study. Technical report, Business Rules Forum, 2005.

[2] D. Hay, A. Kolber, and K. Anderson Healy. Defining business rules - what they really are. final report. Technical report, Business Rules Group, July 2000.

[3] Peter Jackson. *Introduction to Expert Systems*. Addison–Wesley, 3rd edition, 1999. ISBN 0-201-87686-8.

[4] D. Karagiannis. Bpms: Business process management systems. *SIGOIS Bulletin*, 16(1):10–13, August 1995.

[5] Antoni Ligęza. *Logical Foundations for Rule-Based Systems*. Springer-Verlag, Berlin, Heidelberg, 2006.

[6] Sergey Lukichev and Gerd Wagner. Visual rules modeling. In *Sixth International Andrei Ershov Memorial Conference Perspectives of System Informatics, Novosibirsk, Russia, June 2006*, LNCS. Springer, 2005.

[7] Grzegorz J. Nalepa and Antoni Ligęza. Conceptual modelling and automated implementation of rule-based systems. In Tomasz Szmuc Krzysztof Zieliński, editor, *Software engineering : evolution and emerging technologies*, volume 130 of *Frontiers in Artificial Intelligence and Applications*, pages 330–340, Amsterdam, 2005. IOS Press.

[8] Grzegorz J. Nalepa and Antoni Ligęza. A graphical tabular model for rule-based logic programming and verification. *Systems Science*, 31(2):89–95, 2005.

[9] Grzegorz J. Nalepa and Antoni Ligęza. Prolog-based analysis of tabular rule-based systems with the xtt approach. In Geoffrey C. J. Sutcliffe and Randy G. Goebel, editors, *FLAIRS 2006 : proceedings of the nineteenth international Florida Artificial Intelligence Research Society conference : [Melbourne Beach, Florida, May 11–13, 2006]*, pages 426–431, FLAIRS. - Menlo Park, 2006. Florida Artificial Intelligence Research Society, AAAI Press.

[10] OMG. Business process modeling notation (bpmn) specification. Technical Report dtc/06-02-01, Object Management Group, February 2006.

[11] OMG. Semantics of business vocabulary and business rules (sbvr). Technical Report dtc/06-03-02, Object Management Group, 2006.

[12] Ronald G. Ross. *Principles of the Business Rule Approach*. Addison-Wesley Professional, 1 edition, 2003.

[13] W. M. P. van der Aalst, A. H. M. ter Hofstede, and M. Weske. Business process management: A survey. In *Proc. Business Process Management: International Conference, BPM 2003, June 26-27, Eindhoven, the Netherlands*, LNCS, pages 1–12. Springer, 2003.

[14] Barbara von Halle. *Business Rules Applied: Building Better Systems Using the Business Rules Approach*. Wiley, 2001.