The framework for the implementation of business rules in ERP [1]

Oleg Vasilec
Professor, doctor
Vilnius Gediminas Technical University
Saulėtekio al. 11, LT-10223, Vilnius-40
E-mail: olegas@isl.vgtu.lt

Vladimir Avdejenkov
Doctor
Vilnius Gediminas Technical University
Saulėtekio al. 11, LT-10223, Vilnius-40
E-mail: vladimir@isl.vgtu.lt

Sergej Sosunov
Junior researcher, doctor
Vilnius Gediminas Technical University
Saulėtekio al. 11, LT-10223, Vilnius-40
E-mail: sergejus@isl.vgtu.lt

Thes paper presents a model-driven framework for the semiautomatic implementation of business rules in enterprise resource planning systems. The framework is based on the idea of combining business rules represented in natural language with decision tables and triggers in database systems. To support the implementation of the framework, the formal language of templates, which captures patterns of business rules and enables their model-driven transformation to the triggers, is used. This allows a consistent implementation of business rules within ERP systems.

KEYWORDS: *ERP, business rules, templates.*

Introduction

Three main advantages of business-oriented systems are distinguished, (Halle, 2002) such as system agility, reducing the costs, and system transparency. The agility of an information system means that applications can be quickly adapted to the business policy in case of changes. In businesses where the market or other external factors force to change the business policy, one of the goals is zero latency between the confirmation and implementation of a new policy. When organizations approach zero latency, their competitive ability increases. In order to approach zero latency between the confirmation of a policy and the implementation of rules, a modelling tool for business rule management system and business rules must be employed, and it should be available for business analysts to use. When business analysts are able to set the rules, zero latency can be announced as achieved.

[1] The work is supported by the Lithuanian State Science and Studies Foundation within the High Technology Development Program Project "Business Rules Solutions for Information Systems Development (VeTIS)" Reg. No. B-07042.

The employment of business rules can reduce business rule implementation costs significantly, compared with the use of common programming. Such a tendency can be noticed in terms of the time needed to implement the policy: business rule technologies require less human resources and therefore less time for the policy to be implemented. Research shows that IT departments using business rule technologies managed to save 5 to 40% of the budget dedicated to the logic and infrastructural changes.

Rules are logical statements about how a system operates. Some of these rules may be expressed in the business language referring to real-world business entities, and are therefore called business rules. Business rules can represent, among other things, typical business situations such as escalation (”send this document to a supervisor for approval”), managing exceptions (”make sure that we deal with this within 30 min or as specified in the customer’s service-level agreement”). Our conviction is that business rules can be used in the context of ERP.

This paper proposes a framework for business rule transformation that combines business rules, decision tables and triggers. To support the construction of the frameworks, we develop the Business Rules Template Language (BRTL) (Sosunovas and Vasilecas, 2006, 2008), a language of the templates enabling MDA transformations with template representations.

This paper starts with a brief overview of BRTL and its business rules specification approach. Then, it gives an overview of business rules implementation in the ERP framework. Next, it illustrates the framework by applying it to a simple problem. Finally, the paper discusses results of the research presented here, compares it with related work, and makes conclusions.

The BRTL approach

Templates capture the form (or shape) of sentences of some language, generate, upon instantiation, sentences of that language whose form is prescribed by the template, and can be used to describe the commonly occurring structures that make the pattern. Our Business Rules Template Language Formal Template Language (BRTL)(Sosunovas and Vasilecas, 2006) is a language to express templates, which enables transformation of business rules.

A precise notation of the BR template is used for the definition of abstract and domain-specific templates during the design of BR templates as it will be described in the following section. BR templates are not strictly bound to one particular notation. On the contrary, like each model-driven knowledge representation approach, it can have several notations. In this paper, we present one of the possible notations, which will be used for examples in the following chapter.

Each business rule template is constructed from well defined parts called template expressions. Template expressions are separated from each other by template expression metaclass name or short notation keywords and a semicolon (e.g. LiteralExp: or LE:) (Sosunovas and Vasilecas, 2006). The most important template expressions are:

- DeterminerExp (DE) – the determiner for the subject; from the following, the one that makes the best business sense in the statement. Possible values for the name attribute: each, any, etc.;

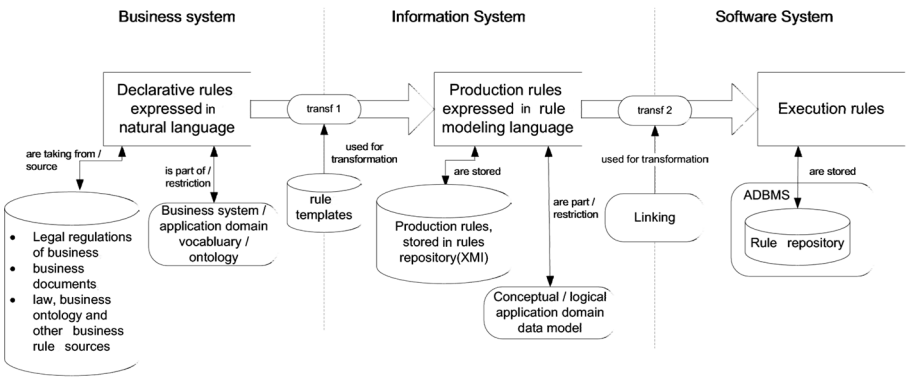

Figure 1. ***General scheme of the method***

- SubjectExp (SE) – a recognizable business entity. The SubjectExp metaclass containsa reference to the ObjectType metaclass from the ORM metamodel;
- CharacteristicExp (CE) – the business behaviour that must take place or a relationship that must be enforced. The CharacteristicExp metaclass refers to the Role and ObjectType from the ORM metamodel;
- LiteralExp (LE) – an instance of the LiteralExp metaclass, containing a character string in the name attribute;
- NumericExp (NE) – an instance of the NumeticExp metaclass, containing a numeric value in the name attribute;

Template expression can have several representations even within the same language (e.g. the verb "to be" may have the presentations "is" or "was"). Template expression from its presentation is separated by a white space, and each presentation is separated from each other by "|" (e.g. LiteralExp: be|is|was).

Three types of template expressions are distinguished: atomic, composite, and reference. Atomic template expressions do not contain other template expressions (e.g. LiteralExp:, KeywordExp:, NumericExp:etc.). Reference template expressions contain references to the other models (e.g. BPMN, ORM). Referenced elements are written after the name of the reference template expression (e.g. SubjectExp: ObjectType: Conference Paper).

Composite template expressions have no special keyword. Different types of brackets are used instead. Parentheses "("and")" are used to enclose a group of mandatory template expressions. Square brackets "["and"]" are used to enclose an optional composite template expression. The number after the second angle bracket denotes the cardinality of the composite template. No number or asterisk "*" sign denotes infinity. Horizontal bars "|" are used to separate alternative template expressions of any type.

Each template expression can be parameterized. Composite template expressions are parameterized by the Template parameters marked with a question mark

after the keyword of the template. It is possible to omit the question mark; in this case, the template expression without presentation will be treated as a parameterized template expression. It is possible to add presentation after the parameterization question mark; in order to provide a parameter example or a default value.

A framework based on business rules management templates

In the following sections, we discuss the main components of the framework proposed by us. The proposed framework (Figure 1) spans three interrelated systems: business system (BS), information system (IS) and software system (SS). According to Caplinskas et al. (2002), the business system of an enterprise to a great extent depends on the supporting information system (IS) and the software system (SS). Therefore, a change in the business system (BS) results in a change of the IS and SS.

The form of business rules varies in the mentioned systems. In particular, in BS, business rules are specified in natural language and are buried in different policies and guidelines in a declarative form. At the same time, it is common practice not to specify business rules explicitly. They can be presented as tacit knowledge of the workers. As for the IS, the formal equivalent of a business rule is an IS law statement that restricts the lawful state space of a thing, as defined in Wand and Weber (1990). From the practical point of view, a business rule in IS should be specified in the form of production rules or at least be explicit. The form business rules take in the SS highly depends on the selected implementation approach.

There are numerous methods used to integrate business rules into information systems (Cibrán, 2002). It can be either an application server implementing business rules, or a business rules managing system itself, which controls and implements a business policy defined by business rules.

Business system

The success of business rules in BS is bound to the way a business user is able to manage them. Business rule (BR) representation as templates is one of the most natural ways to present BR for the business user. The present research is inspired by the success of knowledge representation using templates. Templates have demonstrated their effectiveness in the field of information extraction (Hobbs and Israel, 1994; Sowa, 1999) and specification of ontology axioms (Hou et al., 2002). In both cases they helped to hide complex implementation details and simplify knowledge representation for the user.

A BR template is intended to define the structure of a BR of some particular type. There are gaps left in BR templates to be filled in later when the actual activities of specifying business rules are executed. In addition to automatic processing of BR, another expected advantage of this approach is that users feel comfortable with BR templates as if they were working with natural language statements (Herbst, 1996; Morgan, 2002).

However, it is very difficult to define BR templates acceptable for each enterprise in advance. In particular, it becomes crucial in the worldwide context when adaptation to different cultures and, as a consequence,

languages is a must. Therefore, specification of custom user centric BR templates is necessary. For custom templates to be still available for automatic transformation, they should be built on the basis of well-defined template definition constructs.

One more fact related to the BR template constructs is that BR are rarely used alone, on the contrary, they tend to refer to other enterprise models, in particular, to those of business data and business process. As a consequence, it is necessary to enable BR template constructs to refer to the enterprise models (e.g. object role modelling (Halpin, 1998) and business process modelling notation (BPMN) (Owen, 2003). Therefore, it should be possible to use the components of such meta-models within business rules.

Information system

The main requirement for the representation of business rules in an IS is formality. Business rules should be expressed formally in an IS. The foundation of business rule representation in the information system is laid by the production rules of formal methods. Production rules, which follow the structure

If ... then ...,

are used to express sets of actions and heuristics which can be represented independently of the way they will be used.

In order to process business rules in the form of production rules, we need a technique to store the production rules. If we look at a decision table column by column, a decision table is similar to a set of production rule, and these rules are notoverlapping.

Though originally used as a technique to support programming, decision tables have proven tobe a useful aid in modeling complex decision situations of various sorts. In the field of knowledge-based system research, there has been a renewed interest in decision tables over the past years. The focal point of this approach is the use of decision tables as a modelling formalism for expressing (at least part of) the business knowledge in the problem domain at hand. Decision tables act as an intermediate between the specification of a decision process and the real decision making act, allowing overview and verification throughout the life cycle (Vanthienen et al., 2006).

Rules can be graphically represented in multiple ways for usability purposes. Two structures are primarily used: decision trees and decision tables. A decision table defines a process as an ordered set of conditions and their related actions. Tabular approaches to problem solving were developed in the 1960's by General Electric and other organizations. Official decision table components and terminology were standardized by cooperative efforts of a number of organizations, including the Conference on Data Systems Languages (CODASYL) in 1962. The basic format of decision tables remains largely unchanged today (Cunneyworth, 1994).

The decision table formalism is not an isolated technique and shows a lot of interfaces to other representation formalisms such as code, trees, rules, etc. Making good use of these connections, however, is only possible through a flexible computer support. Therefore, a variety of bridges have been built between the decision table workbench and other representations, resulting

3	TriggerEvent Before Insert StockTransaction
4	checkCreditLimit(Customer Code; Transaction Type; Order Number);
5	
6	TriggerEvent Before Insert StockTransaction
7	checkCreditTerm(Customer Code; Transaction Type; Order Number);
8	
9	TriggerEvent After Update Customer
10	checkCustomerBlock();

Figure 2. ***Business rule event description***

22	Rules void checkCustomerBlock(Customer Code)				
23	C1	C1	A1	A2	
24	checkCurrentCustomerStatus(Customer Status)	checkPreviouslyCustomerStatus(Customer Status)	sendMail(Customer.Customer e-mail)	sendMail(Order.Salesman e-mail)	
25	int	int	boolean	boolean	
26	**Current Customer Block**	Previous **Customer Block**	**Send Mail to Customer**	**Send Mail to Customer**	
27	1	0	TRUE	TRUE	
28	0	1	FALSE	FALSE	
29					
30	Rules void checkCustomerUnBlock(Customer Code)				
31	C1	C1	A1	A2	
32	checkCurrentCustomerStatus(Customer Status)	checkPrevious CustomerStatus(Customer Status)	sendMail(Customer.Customer e-mail)	sendMail(Order.Salesman e-mail)	
33	int	int	boolean	boolean	
34	**Current Customer Block**	Previous **Customer Block**	**Send Mail to Customer**	**Send Mail to Customer**	
35	1	0	FALSE	FALSE	
36	0	1	TRUE	FALSE	
37					
38	Rules void checkCustomerUnBlock(Customer Code)				
39	C1	A1	A2	A3	

Figure 3. ***Business rule decision table***

in a large application domain for decision table modelling (Cunneyworth, 2004).

When building a decision table, the designer essentially provides the system with the following information: a list of conditions with their states, a list of actions and a list of rules.

The analysis of business rules management systems (QuickRules, Oracle Business Rules, Versata, Ilog Rules and Open Rules) under different aspects (ILOG, 2001; Yasu Technologies; Versata Inc.; Open Rules) showed that decision tables and production rules are the main approaches to enter business rules. At the same time, the method of the integration of rules into information systems using triggers is not offered in those systems. This is so because usually business rule management systems are oriented towards the newly created systems. When creating a system, other ways to realize business rule are anticipated (e.g. application servers, objects, plug-ins, etc.). Furthermore, an analysis of business rule management systems was performed considering the potential to be connected to BRMS depository for the further use of rules.

The business rule management system chosen for an experiment is OpenRules. It

is an open source system with the interfaces for creating rules and is easily comprehensible not only for technical specialists, but also for business people. In order to have a possibility to describe triggers with the help of OpenRules, a new keyword, Trigger Event was entered into the OpenRules system. This extension analyses the described business rules, generates the SQL trigger code, and installs triggers into a database management system of SS.

Software system

One of the ways to realize business rules in software systems is active database management systems (Avdejenkov and Vasilecas, 2007; Valatkaite and Vasilecas, 2003) . Active database triggers the use of the ECA rules model and consists of three parts: an event, a condition and an action, abbreviated as ECA. An event marks the moment when the rule comes into effect, a condition is screened when the rule begins to operate, and an action is performed when an event comes into effect and the condition is correct. If a set event takes place, a condition is screened; this could be either a screening of installed or deleted data, or a screening of data in other charts, etc. Depending on the results of condition screening, it could be any kind of SQL language command: a cancellation of a transaction, changes to the data either in the base chart or any other chart, a dispatch of messages, etc.

As we see, the employment of triggers can ensure a wide functionality of a business management system. As an example, the study of Avdejenkov and Vasilecas (2007) analyses a solution to the debtor management task using triggers. Debtor management is marginally automated in business management systems; moreover, conditions and business policy regarding it may change rapidly, therefore the employment of rule at this point would ensure all benefits of business rule management

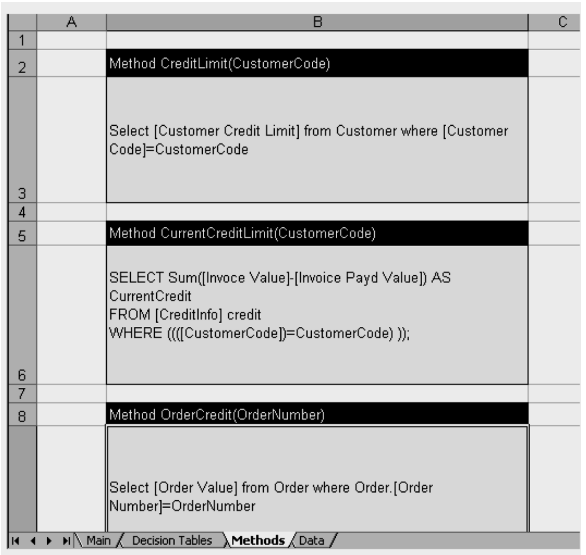

Figure 4. ***Methods of business rules***

(business rules could be changed quickly, debtor management would be transparent, maintenance costs would be reduced). In the study, it is suggested to design the rules using UML diagrams. The rules designed are transformed into DBMS triggers and integrated into a business rules management system.

Experimental validation of the framework

In order to evaluate the proposed framework of implementation of business rules into the ERP system, we have selected "Scala" as a business management system and debtor management as a business scope of the enterprise resource planning system. The choice of the enterprise resource planning system was determined by the fact that this system allows to perform real trials and to obtain tangible results which can be assessed. Also, this enterprise resource planning system is well known worldwide. Its clients are based in 140 countries, with the number of clients exceeding 20 thousand.

The problem area chosen for this paper is debtor management in a trade company. This choice was determined by the importance of automated debtor management to the majority of companies, and, on the other hand, ERP systems do not always properly automate this process. In the chosen ERP system "Scala", debtor management is not properly automated, either.

1. Block the delivery of goods to the customer, if his credit limit is exceeded after a trading operation.
2. Block the delivery of goods to the customer, if his payment is 3 or more days late.
3. Allow the customer to make the purchase, if his payment is not 3 or more days late.
4. Send a reminder e-mail to the customer and the sales manager working with the customer, if the delivery is blocked.
5. Send a reminder e-mail to the customer and the sales manager working with the customer, if the delivery is unblocked.

From the presented rules it is possible to specify two business rule templates using the BRTL. The first one addresses rules 1–3 and the second one rules 4–5.

The BRTL rule template for the rule *Block ... if ...*:

(KE "Block"|KE "Allow") (
SE "the delivery of goods to the" CE "customer"
|KE "the" CE "customer" KE "to make the purchase")
KE ", if" KE "his" (
(CE "credit limit" CE "is exceeded"
KE "after the trading operation.")
| (CE "payment" KE "is" [KE "not"] NE ?
KE "or more days late."))

This rule's user view:

(Block | Allow)
(the delivery of goods to the customer| the customer to make the purchase)
, if his (
(credit limit is exceeded after the trading operation.)
| (payment is [not] { ? } or more days late.)
)

The second BRTL template to enter e-mail sending to the sales manager rules:

KE "Send" SE "a reminder e-mail" KE "to " CE "the customer" KE "and" CE "the sales manager" CE "working with" CE "the customer" KE "," KE "if" CE "the delivery" (CE "is unblocked"| CE "is blocked")

This rule's user view:

Send a reminder e-mail to the customer and the sales manager working with the customer, if the delivery (is unblocked | is blocked)

In order to encode ECA type rules (for later transformation into ADBMS triggers), we have extended the OpenRules notation with an additional keyword TriggerEvent, followed by the keywords BEFORE or AFTER describing the time when the rule is coming into effect, followed by the keywords INSERT, DELETE or UPDATE describing the event for the business rule to react (Figure 2).

After the event description, it is denoted which table the business rule is attributed to. The next row in the table includes a link into the OpenRules decision table describing conditions to be checked and actions to be taken as subject to the result of condition check (Figure 3). Both the link to the decision table and the decision table match the ordinary OpenRules notation. Decision tables include links to the methods (Figure 4) that are implemented in trigger actions depending on the results of conditions later on.

After the event description, it is denoted which table the business rule is attributed to.The next row in the table includes a link into the OpenRules decision table describing conditions to be checked and actions to be taken as subject to the result of condition check (Figure 3). Both the link to the decision table and the decision table match the ordinary OpenRules notation. Decision tables include links to the methods (Figure 4) that are implemented in trigger actions depending on the results of conditions later on.

With the help of OpenRulesToSQL extension, we have constructed the final triggers, inserted them into the database management system of the Scala ERP system and tested the correctness of operations in the ERP system. After chechking the functionality of the ERP system, it was launched in a usual production mode, with newly produced and integrated business rules representing new business logics needed.

The main advantage of the proposed approach is that business rules are separated from the ERP system code, so that rule parameters, such as deviations from the credit limit overdrafts or the allowed extension of a payment deadline, may be changed directly by business people as well.

Within two months after implementation of debtor management business rules, as a result of experimental validation of the method, we improved the situation with debtors in the company where this test had been performed:

- debt turnover decreased from 70 to 60 days;
- debts decreased by 15
- bad debts decreased by 25

Summarising the experimental validation, we have proved the method proposed and the possibility of business rule involvement, using the business rule management system, into the DBMS of Scala ERP system. By doing this, we assumed that Scala ERP has a trivial client/server architecture with separate DBMS commonly used by contemporary ERP systems. Therefore, we suppose the possibility of a similar use of business rules represented as ECA rules for implementation of new logics also in other types of ERP systems.

Related works

A work that is close to ours is Catalysis (D'Souza and Wills, 1998), a modelling method based on the UML. Like Catalysis, we also use the idea of model frameworks and templates to make models of reusable assets, the ideas of model refinement with UML diagrams, and the idea of defining the semantics of UML constructs adapted to the context in which they are used. Our approach, however, develops these ideas within a formal framework. The template notation used in the Catalysis has less features than the BRTL (only parameters; BRTL has choice and lists).

Vasilecas and Smaizys (2007) report on the use of decision tables as an alternative business logic visualisation formalism used for business rule representation and further transformation using XML. The parameterisation task of the SS is solved by automatically building the SS configuration parameter set and executable components of n-Tier architecture according to the set of rules in decision tables, previously captured by a business domain analyst.

There are two main views in dynamic business rule driven software system design. One of them is to design predefined executable processes and execute them by using rules in the software system, where processes and execution rules are derived from business rules using transformations (Orriëns et al., 2003). The other is discussed in our papers (Vasilecas and Smaizys, 2006), where business rules and facts describing the current business system state are loaded into the inference engine of the software system and transformed into a software system executable data analysis process according to the results of logical derivations.

Conclusions

This paper advocates an approach to integrate business rules to ERP using transformations. To support this approach, the paper presents a language of business rules templates (BRTL) that enables transformation of business rules. BRTL is used to construct catalogues of business rules (expressed as templates), so that instances of business rules are generated by instantiating templates. The paper also presents and illustrates a framework for the implementation of business rules in ERP, which combines the declarative, production and ECA rules.

REFERENCES

V. Avdejenkov and O. Vasilecas. Business rules applying to credit management. In Khaled Elleithy, editor, *Advances and Innovations in Systems, Computing Sciences and Software Engineering*, chapter Business Rules Applying to Credit Management, pages 481–483. Springer, 2007.

A. Caplinskas, A. Lupeikiene, and O. Vasilecas. Shared conceptualisation of business systems, information systems and supporting software. In H.-M. Haav and A. Kalja, editors, *Databases and Information Systems II, Selected Papers from the Fifth International Baltic Conference, BalticDB&IS'2002*, pages 109–320. Kluwer Academic Publishers, 2002.

M. A. Cibrán. Using aspect-oriented programming for connecting and configuring decoupled business rules in object-oriented applications. Master's thesis, Vrije Universiteit Brussel, Belgium, 2002.

W. Cunneyworth. *Table Driven Design–A Development Strategy for Minimal Maintenance Information Systems*, 1994. URL: http://www.dkl.com/documents/pdf/table\s\do5(d)riven\s\do5(d)esign.pdf.

W. Cunneyworth. *Adaptable Design Helps Business Practices Become Automated Practices*, 2004. URL: http://www.dkl.com/documents/pdf/adaptable\s\do5(d)esign.pdf.

D. F. D'Souza and A.C. Wills. *Objects, components, and frameworks with UML: the catalysis approach*. Addison-Wesley Longman Publishing Co., Inc. Boston, MA, USA, 1998.

Barbara Halle. *Business Rules Applied*. Wiley, New York, 2002. ISBN 9780471412939.

Terry Halpin. Object-Role Modeling (ORM/NIAM). In *Handbook on Architectures of Information Systems*, pages 81–102. Springer-Verlag, 1998.

H. Herbst. Business rules in systems analysis: A meta-model and repository system. *Information Systems*, 21(2):147–166, 1996.

J. Hobbs and D. Israel. Principles of template design. In *Arpa Workshop on Human Language*, pages 172–176, 1994.

C. S. J. Hou, N.F. Noy, and M.A. Musen. A template-based approach toward acquisition of logical sentences. In *Intelligent Information Processing: Ifip 17th World Computer Congress-Tc12 Stream on Intelligent Information Processing, August 25-30, 2002, Montreal, Quebec*. Kluwer Academic Publishers, 2002.

ILOG. *ILOG Solver 5.1 User's Manual*. ILOG s.a., 2001. URL: http://www.ilog.com.

T. Morgan. *Business Rules and Information Systems: Aligning It With Business Goals*. Addison-Wesley Professional, 2002.

Open Rules. *Open source business rules management system*. URL: http://openrules.com.

Bart Orriëns, Jian Yang, and Mike P. Papazoglou. A framework for business rule driven service composition. *Lecture Notes in Computer Science*, 2819:14–27, 2003. URL: http://springerlink.metapress.com/openurl.asp? genre=article&issn=0302-9743-&volume=2819&spage=14.

J. Owen M., Raj. *BPMN and Business Process Management, Introduction to the Business Process Modelling Standard*. Popkin Software, 2003.

Sergejus Sosunovas and Olegas Vasilecas. Precise notation for business rules templates. In *Databases and Information Systems, 2006 7th International Baltic Conference*, pages 55–60, 2006.

Sergejus Sosunovas and Olegas Vasilecas. Practical application of BRTL approach for financial reporting domain. *Information Technologies and Control*, 37 (2):106–113, 2008.

J. F. Sowa. Relating templates to language and logic. *Lecture Notes In Computers Science*, 1714:76–94, 1999.

Irma Valatkaite and Olegas Vasilecas. A conceptual graphs approach for business rules modeling. *Lecture Notes in Computer Science*, 2798:178–189, 2003. URL: http://springerlink.metapress.com/openurl.asp? genre=article&issn=0302-9743-&volume=2798&spage=178.

J. Vanthienen, C. Mues, S. Goedertier, R. Laleau, and M. Lemoine. Experiences with modeling and verification of regulations. In *International Workshop on Regulations Modelling and their Validation & Verification (REMO2V '06), in conjunction with the 18th Conference on Advanced Information System Engineering (CAiSE '06)*, Luxembourg, June 2006. Presses Universitaires de Namur.

O. Vasilecas and A. Smaizys. Business rule based data analysis for decision support and automation. In *Proceedings of 7th International Conference on Computer Systems and Technologies – CompSysTech'06*, pages II.9–1–II.9–6, 2006.

Olegas Vasilecas and Aidas Smaizys. Business rule based software system configuration management and implementation using decision tables. In Yannis E. Ioannidis, Boris Novikov, and Boris Rachev, editors, *ADBIS Research Communications*, volume 325 of *CEUR Workshop Proceedings*. CEUR-WS.org, 2007. URL: http://ceur-ws.org/Vol-325/paper03.pdf.

Versata Inc. *Versata 6. Product Documentation*. URL: http://kb1.versata.com/default.asp?id=1614&Lang=1.

Y. Wand and R. Weber. An ontological model of an information system. volume 16, pages 1282–1292, 1990.

Yasu Technologies. *Business Rule Management Suite*. URL: http://www.yasutech.com/products/index.htm.

KARKASAS, SKIRTAS ĮGYVENDINTI VERSLO TAISYKLES VERSLO VALDYMO SISTEMOJE

Oleg Vasilec, Vladimir Avdejenkov, Sergej Sosunov

S a n t r a u k a

Straipsnyje pateikiamas modeliais grindžiamas informacinių sistemų kūrimo karkasas, skirtas automatizuotai įgyvendinti verslo taisykles verslo valdymo sistemose. Straipsnyje siūlomas karkasas sudarytas iš komponentų, kuriuose verslo taisyklės pateiktos natūralia kalba, sprendimo lentelėmis ir taisyklėmis, įgyvendintomis trigeriais duomenų bazėje. Siekiant įgyvendinti siūlomą karkasą, straipsnyje pateikiama formali šablonų užrašymo kalba, skirta automatizuoti verslo taisyklių įvedimą. Ši kalba leidžia keisti įvestas verslo taisykles į verslo valdymo sistemos vykdomą programų sistemos kodą. Taip yra užtikrinamas darnus verslo taisyklių įgyvendinimas verslo valdymo sistemoje.