EXTRACTING BUSINESS RULES – HYPE OR HOPE FOR SOFTWARE MODERNIZATION

Neli Maneva, Krassimir Manev

***Abstract**: The purpose of this work is to investigate how the Business Rules approach (very promising) can be used for the purposes of feasible and efficient software modernization (very needed). The processes of the forward and backward reengineering of legacy software systems are presented through identification of the main players. Their responsibilities, assigned tasks and expected deliverables are described. Some reasons for need of instrumental tools to support the process are given.*

***Keywords:** software development, software modernization, business rules, extracting business rules.*

***ACM Classification Keywords:** D.2.1 Requirements/Specifications, D.2.7 Distribution, Maintenance, and Enhancement.*

Introduction

According to the statistical data, since 2009 more than 80% of software applications are "remake" of already existing, but older or obsolete applications [Jones, 2010]. Such *legacy software systems* [Dictionary, 2012] still have business importance, but are expensive to manage and difficult to modify, with poor competitiveness and compatibility with modern equivalents. This explains the current intensive research and development work on software modernization. Experts, involved in software modernization, have borrowed a number of general and validated Software Engineering (SE) methods and tools, but in order to solve specific problems, they create their own approaches [Comella-Dorda, 2000]. For example, to make some unavoidable changes and to meet new requirements, the achievements of Requirements Engineering theory and practice can be used as a base, but should be applied within a modified process, because of insufficient and/or lost expert's knowledge. The missing information (including algorithms and business rules) should be extracted from selected legacy system artifacts and together with the identified new requirements, should be built into the modernized application.

The paper is devoted to some problems of introducing a Business Rule (BR) approach to a specific software engineering activity, namely software modernization [Hay&Healy, 2000]. Section 2 describes the main players and their roles within a BR–based software development, which can be considered as a proactive approach, facilitating further maintenance and modernization of the developed software system. In Section 3 the conception

of the BR-based software modernization as reverse engineering is presented. Some perspectives for computer-aided support to a number of the considered activities are briefly commented in Section 4. In Conclusion the pros and cons of the proposed approach are summarized and a few ideas for future research and development work are mentioned.

BR-based software development

Development of software with usage of business rules is very close to the traditional one – just much more attention is dedicated to the rigorous analysis of the enterprise policy. As a result of this analysis a set of business rules is constructed. Those of business rules, influencing in any way some aspects of the software system under development, should be identified, separated in a group and further processed as special (BR-generated) system's requirements. For better understanding, the process of elicitation and analysis of such requirements can be presented through the work flow and data exchange among the involved participants. Using the classification of WebSphere ILOG BRMS of IBM [Stineman, 2009], we rank the distinguished 5 main roles of people, taking part in the BR-based development of a software system, as follows:

- Business Analyst – responsible for domain modeling and preparing the business rules vocabulary;
- Policy Manager – translates the business policy into detailed BR. For our purposes the Policy Manager is representative of the business people who (following the *Zachman framework)* are involved in the process of creating the business application [Zachman International, 2008];
- Software Architect – responsible for software system architectural design;
- Software Developers – create and test the application, iteratively adding new functionality, as required;
- System Administrator – observes and controls the running application in order to achieve the stated goals.

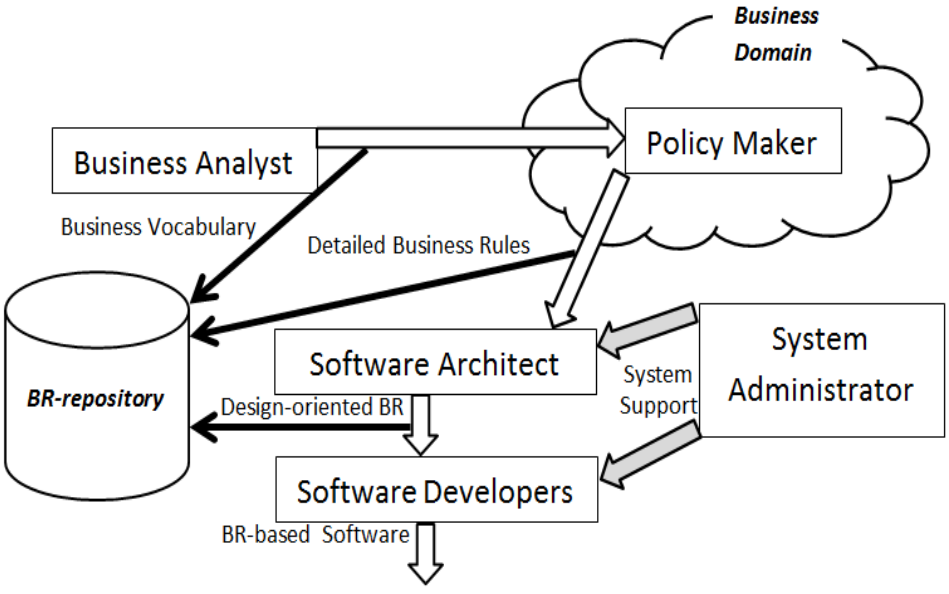

Fig. 1. BR-based software development

In Fig.1 the double arrows demonstrate the main work stream of the forward process of software development. The single arrows show which deliverables, obtained as results of a participant's activity during the different steps of process, will be saved in the BR-repository.

Next follows a brief description of responsibilities for two roles, specific for BR processing.

Business analyst is a person with an appropriate background, qualification and experience, who supplies the vocabulary of the business domain. We assume that in general (s)he should be an expert, hired as an external consultant for the project under consideration. The independence of the business analyst has to guarantee that the vocabulary is as complete as possible and acceptable for each business subject in the domain.

Policy Maker of the enterprise formulates the management policy of a given enterprise and expresses it in a set of business rules with the desired level of details. The Vocabulary and the set of business rules, represented by appropriate formalism (see for example [Hay&Healy, 2000]), are saved in the repository and can be used during the detailed design of the enterprise's software system.

After this initial phase of requirements definition (including BR-generated ones) the further development process can be quite similar to the traditional one, slightly modified in order to reflect some peculiarities, imposed by the business rules [Andreescu, 2009].

BR-based software modernization – BR extraction from the software system

Modernization of a legacy software system with usage of business rules could be split in three major stages – Extracting the built in the legacy software business rules, Modernization of the extracted rules and Re-design and implementation of the system on the base of modernized business rules. Obviously, the most challenging of these three stages is the first one. That is why we will concentrate on it. The Fig. 2 schematically shows one possible way of process organization through description of the roles of the participating people and the created deliverables.

It is clear that the role of the ***Business Analyst*** will be changed a bit during the process of BR extraction from the legacy software system. Preparing of adequate and acceptable business vocabulary will not be enough for successful extraction of business rules. As the classification of BR in [Hay&Healy, 2000] suggests, instead of vocabulary, an *ontology of the business domain* has to be applied. Here we will use the term *ontology* as it is understood in computer science and information technologies, i.e. an ontology is consisting of "the objects, concepts, and other entities that are assumed to exist in some area of interest and the relationships that hold among them" [Genesereth&Nilsson, 1987]). In such a way mapping of candidate business rules, extracted from the legacy system, to sentences in the language of the business people has to be much more easy.

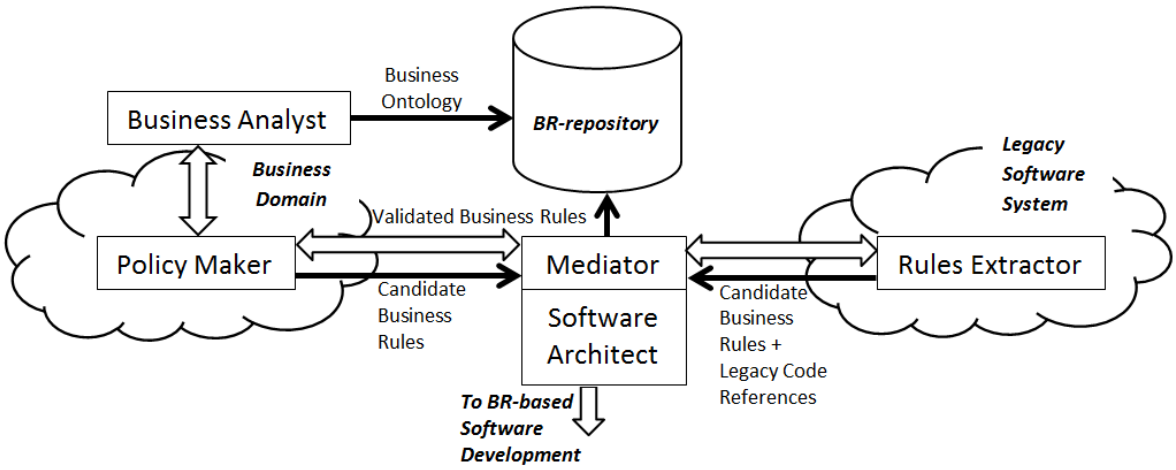

Fig. 2. Extracting BR from Legacy Software System

Nowadays there are well developed and validated ontologies of many domains that are actively used in *knowledge-based* (or *expert*) *systems*. Usage of ontologies became even more popular with introducing the *semantic Web* concept some years ago and so there is very high level expertize in this area. That is why it will be relatively easy to find or elaborate the necessary ontology for the BR-based modernization of software systems in an arbitrary area. By the way, usage of ontology instead of a simple vocabulary of the domain will be much more helpful in the design and development of BR-based software system when starting their development from scratch.

If the owners of the business would like to have a modern BR-driven software system, they have sooner or later to describe their main business policy in a form of detailed business rules. Such task is intrinsic for the ***Policy Maker*** who in our concept is a representative of the owners. His role in the process of BR extraction from a legacy software system is double. From one side, the role will be practically the same as in the process of BR-development from scratch. The Policy Maker is translating the business policy of owners into business rules again, using formal description and the ontology of the domain. But from the other side, the Policy Maker has to try to define as more as possible candidate business rules, which, it is supposed, are implemented in the legacy software system. For making our concept clearer, it will be better in this case to rename the role of Policy Maker to ***Policy Translator***.

An important new role in the process of BR extraction from a legacy software system will be the ***Rules Extractor***. His task is to mine available software system resources (source code, executables, documentation, data bases, etc.) and to try to formulate some candidates for business rules, incorporated in the software system. These rules will be expressed in the terms of the software system (variable and function names, attributes of data base tables,

caption of elements in screen forms, etc.) but with the same formalism, used by the Policy Translator. It is very important that to each candidate rule, extracted from the system, there will be an assigned respective reference to the software system. In such a way, if the extracted rule has to be changed in the modernized version of the system, Software Developers will be able to localize the component of the system to be rewritten.

Even more important new role in the process of BR extraction from a legacy system is the ***Mediator***. Let *A* be the set of candidate rules specified by the Policy Translator and *B* be the set of candidate rules, extracted from the system by the Rules Extractor. The task of the Mediator will be to find a candidate rule *p* from *A* and a candidate rule *q* from *B,* which seems to be adequate. Let r_q be the reference assigned to *q* by the Rules Extractor. We will call the couple (p, r_q) *extracted rule.* As many rules are extracted from the legacy system, as easy will be the work of Mediator during the next stage of modernization, does not matter how the process will be performed – with rewriting parts of the system or from scratch.

The work of the Mediator could not be successfully done without the help of people who know in detail the legacy code and platform. In the Fig. 2 this role is denoted by Software Architect with the clear understanding that there is a team of people that stay behind it. In practice this will be the IT team of the enterprise that exploits and maintains the system and we will call it ***the team of Software Architect***. It is quite possible some people from the team of Policy Maker to be involved in activities from this stage of the extraction process, too.

When the process of BR extraction from the legacy system is completed, the extracted rules have to be checked for compliance with the new requirements to the enterprise software. Some of the extracted rules will **contradict the new requirements**. The elimination of these rules will be crucial for the design of the modernized system. If it is possible to eliminate the part of the system that implements these rules then the modernization could be made by introducing some changes. Else, the system probably has to be rewritten from scratch.

If the system will not be rewritten from scratch then it is worth to outline the extracted rules that **deviate from the new requirements.** This will give the team of Software Architect the possibility to identify the parts of the legacy system which will be affected by modernization. Finally, a set of these BR is composed that meets to a satisfying degree the new requirements to the enterprise system. Then the process could continue as in the case of the normal BR-based software development.

We would like to stress a major difference between the two processes. The BR-based design and development is almost a linear process and the outlined steps of this process are executed consecutively. The BR-based modernization of a legacy software system, as demonstrated above, is much more sophisticated from managerial point of view. In some cases it will be very difficult to organize the process properly - especially its part, dedicated to BR extracting from the system through a clearly defined tasks sequence. The opposite, our expectations are that different activities of the extraction will be performed in parallel, mutually challenging and influencing each other.

How to automate the BR extraction from a legacy system

As it was shown above, the process of BR-based modernization of legacy software system comprises a number of nontrivial tasks. It is difficult to imagine that these tasks could be solved manually. The main goal of this paper is to investigate the process of legacy system modernization so as to outline the tasks or subtasks, for which some special-purpose software tools will be necessary. As the resources that the legacy software system provide in order to help the process of modernization will be the inputs of such software tools, let us first identify the most important of them (in decreasing order of their importance):

Source code of the legacy system – this is the **most valuable resource** of the legacy system for the BR-based modernization process, even if it is written in an exotic programming language and for an exotic platform. Being a totally formal object, the source code is the most appropriate for automated mining. Something more, many software tools have been developed for years with different purposes – calculating software metrics, transparent-box software testing, etc. Some well developed methods, as **static** and **dynamic analysis**, could be tuned for the purposes of the BR extraction from code. Manual mining in the source code of the legacy system could be necessary in some very specific cases, e.g. when a formal grammar or any other formal description of the used language is missing, as well as there no available working compiler for checking some hypothesis about unknown elements of the syntax.

Executable modules of the system could be the most valuable resource, when the system's **source code is not available,** but if and only if the ***platform,*** necessary to run executables is available. Executables could be used in two ways. First, to use them without modifications, i.e. as they are. This is very close to the usage of executables in nontransparent (or black box) software testing. Such approach is also well developed and could be adapted for BR extraction. The second possibility to use executables is to reassembly them and to try to use the obtained result as a source code. No doubt, extraction of rules from code, written in assembly language, will be more difficult and algorithmically different from rules extraction from code, written in high level programming language. In this case it is worth to consider translation from assembly language to a very close to assembly high level programming language (kernel C, for example). Even better results have to be expected when the extracting process is **performed in parallel** on the source code and on the executables.

Data base of the legacy system could be valuable source for extracting rules. Even not working data base could be helpful with its data base scheme and the formal description of the tables, especially when the attributes of the records have some mnemonic names. Rules extraction from the structure of the base is relatively easy and could be done manually or with a simple program. More interesting and more difficult task will be extracting rules from SQL scripts. Unfortunately, the approaches for rules extraction from high level programming languages will be probably unusable for SQL scripts.

As it was shown, the three above mentioned resources are suitable for different by hardness and used algorithms automation. The corresponding programs will form the toolkit of the agent, called Rules Extractor. From the other

side of the process – those of Police Maker – automation is almost impossible. The candidate rules that Police Maker will propose will be extracted manually from two sources. The first of them is the ***knowledge of the staff*** about the legacy software that has to be collected and formulated as candidate rules by interviews with all level staff, but especially with the people that exploit and maintain the software.

The other resource that has to be explored by the team of Policy Maker is the ***system maintenance documentation*** and ***user manuals*** of the legacy system (if they exist and are with good quality). Some automation here is also possible – there are examples of intelligent text processors that could be tuned to search in documentation in digital form. We think that this work requires intelligence and special competences and will be done rather manually.

The final stage of BR extraction covers the tasks of the Mediator's team. Suppose that the two set of candidate rules are expressed by the same formalism and the difference is that the first set uses the words of the ontology and the other – the mnemonic names of the implementation. Then a mapping of mnemonic names into the terms of the ontology has to be done in order to obtain a set of business rules and each rule is assigned to the place(s) in the software where it is implemented. It seems that this activity could be automated, using the knowledge and experience of the researchers in the domain of ontologies generation and maintenance.

Conclusion

The paper presents an initial study how the BR approach can be integrated into the process of software modernization. The main contribution is that the BR-related activities are described in the context of working environment, in which the participants with predefined roles are responsible for a set of dynamically assigned tasks, accomplished through participatory teamwork.

A number of hard problems, hampering the real life implementation of the proposed approach, can be seen even in this initial stage of research. For example, it will be difficult to select the most appropriate presentation and adequate transformation of the BR, shared among participants with quite different background and decision-making status. There are some very hard pure technological problems, e.g. how to organize the BR repository so as to assure both easy and fast access to the saved items, etc.

We intend to continue the research, trying to combine the BR approach with the described in [Maneva&al., 2011] model-based software modernization.

Acknowledgments

This work is supported by the National Scientific Research Fund of Bulgaria under the Contract ДТК 02-69/2009.

Bibliography

[Andreescu, 2009] A. Andreescu. A General Software Development Process Suitable for Explicit Manipulation of Business Rules. In: International Conference on Computer Systems and Technologies – CompSysTech'09, 2009.

[Comella-Dorda&al., 2000] S. Comella-Dorda, K. Wallnau, R. Seacord and J. Robert. A Survey of Legacy System Modernization Approaches. Software Engineering Institute, Carnegie Mellon University, Pittsburgh, 2000.

[Dictionary, 2012] Legacy System, http://dictionary.reference.com/browse/legacy+system, visited 15.02.2012.

[Genesereth&Nilsson, 1987], M. R. Genesereth, N. Nilsson. Logical Foundations of Artificial Intelligence. Morgan Kaufmann Publishers: San Mateo, CA, 1987.

[Hay&Healy, 2000] D. Hay and K. A. Healy (eds.). Defining Business Rules ~ What Are They Really? GUIDE Business Rules Project Final Report, rev. 1.3., July, 2000.

[Jones, 2010] C. Jones. Software Engineering Best Practices. Mc Grow Hill, 2010.

[Maneva&al., 2011] N. Maneva, Kr. Kraychev and Kr. Manev. Mathematical Models-Based Software Modernization. In: Mathematika Balkanika J., New Series, vol.25, No 1-2(2011), pp. 131-144.

[Stineman, 2009] B. Stineman, IBM WebSphere ILOG Business Rule Management Systems: The Case for Architects and Developers. IBM Software Group, November 2009.

http://www-01.ibm.com/software/websphere/products/business-rule-management/, visited 15.02.2012.

[Zachman International, 2008] The Zachman Framework: The Official Concise Definition. Zachman International, 2008.

http://zachman.com/about-the-zachman-framework, visited 15.02.2012.

Authors' Information

Neli Maneva *– Associated Professor, Ph.D., Institute of Mathematics and Informatics, Bulgarian Academy of Sciences, Acad. G. Bonchev str., bl. 8, Sofia 1113, Bulgaria; e-mail: neman@math.bas.bg*

Major Fields of Scientific Research: Software Engineering, Software Quality Assurance, Model-driven software development.

Krassimir Manev *– Associated Professor, Ph.D., Department of Mathematics and Informatics, Sofia University, 5 J. Bourchier blvd., Sofia-1164, Bulgaria; e-mail: manev@fmi.uni-sofia.bg*

Major Fields of Scientific Research: Discrete mathematics and Algorithms, Formal Methods in Software Engineering, Source Code Analysis, Software Testing.