INFORMATICA, 2006, Vol. 17, No. 4, 503–518

Repository for Business Rules Based IS Requirements

Kęstutis KAPOČIUS, Rimantas BUTLERIS
Department of Information Systems, Kaunas University of Technology
Studentu 50, LT-51368 Kaunas, Lithuania
e-mail: kestutis.kapocius@ktu.lt, rimbut@if.ktu.lt

Received: October 2005

Abstract. The quality of software engineering projects often suffers due to the large gap between the way stakeholders present their requirements and the way analysts capture and express those requirements. With this problem in mind the new method for business rules driven IS requirements specification has been developed. In this paper the architecture of the requirements repository, which is at the core of the proposed method, is presented. The repository model supports the storage and management of all components of the captured requirements, including functions, business decisions, data sources, conceptual data model elements, business rules and their templates. The important aspects of the specialised requirements specification tool implementation are also overviewed.

Key words: business rules, information system, repository, requirements specification.

1. Introduction

One of the fundamental issues concerning the improvement of Information systems (IS) requirements specification process is the reduction of the gap between business representatives and analysts. Because of such division requirements may be lost during specification, while specification itself may be difficult to verify for the stakeholder. A relatively complicated representation of business knowledge and especially business rules (BR) is often considered as the main cause of the problem. This leads to the assumption that IS requirements should be more BR oriented. With this view in mind the framework for BR driven IS requirements specification has been developed (Kapocius *et al.*, 2005) and it soon evolved into a new method. The method is based on structuring of natural language business rules that are discovered analysing business functions and underlying business decisions. The proposed requirements specification process offers the following benefits:

1. Increased consistency of requirements. This is achieved due to the rigid structure of discovered information, which is managed using a single requirements repository system. The repository could also help to achieve a fluent transition from the requirements analysis to the design phase of IS development.
2. Increased requirements verification and modification efficiency. Obviously, natural language rules are easier to understand and verify for stakeholders.

3. Structuring of business rules in a repository is useful as it allows highlighting the autonomous functions of the system and their interrelations. That may help during the development of multi-component systems (e. g. ones that provide Web-services).

This paper presents the architecture of the repository for the requirements elicited using the aforementioned requirements specification approach. Development of the repository facilitated the creation of the new requirements specification method and therefore analysis of its architecture can be considered as the overview of the method itself. The brief introduction into the subject is presented in Section 2. Section 3 contains short overview of the proposed requirements specification process. In Section 4 the architecture of the requirements repository is introduced, its implementation is discussed in Section 5.

2. BR-Oriented Requirements Specification: General Overview

According to the GUIDE Business Rules Project, business rules can be derivations, structural assertions or action assertions (constraints) (Hall *et al.*, 2000). Structural assertions can be either terms or facts that describe the relationships between those terms. Basically structural assertions describe data requirements while action assertions and derivations cover various process-related and other requirements. The majority of BR researchers, including us, adopt such view.

Our proposal is based around the idea that natural language business rules can be the basis for IS requirements (Kapocius *et al.*, 2005). One of the better-known methodologies based on this assumption is TEMPORA (TEMPORA Consortium, 1993). The BR-oriented requirements specification has also shown its applicability in the case of programming centred paradigm, such as Extreme Programming (XP), where it was shown that requirements could be gathered in the form of BR-based Extreme Requirements (Leonardi *et al.*, 2002).

A similar research but in the area of only structural BR modelling proposed Agent-Object-Relationship diagrams as an agent-oriented extension of Entity-Relationship diagrams, with the possibility to express active and passive entities in order to capture more semantics of the dynamic aspects of organizations (Taveter *et al.*, 2001). However, this work was oriented toward the area of business interaction processes modelling and did not cover the issues of IS design process improvement.

Some of the BR modelling approaches suggest the use of Object Constraint Language (OCL) as a tool for an initial definition of business rules (Demuth *et al.*, 2001). Indeed, OCL provides the possibility to deal explicitly and automatically with business rules when building UML-based applications. However, the syntax of OCL is very formal and technical. Therefore OCL is hardly useful in writing down requirements, which have to be presented to the business-owner for evaluation (Maciaszek, 2001).

An interesting ongoing research on BR structuring is MBRM (Manchester Business Rules Method), which incorporates a so-called *Link* business rules model (Kardasis *et al.*, 2004; Wan-Kadir *et al.*, 2003; 2005). In MBRM the conceptual specification of rules (based on natural language templates) can be linked directly to software designs.

Barbara von Halle presented a particularly exhaustive methodology of BR-driven IS development in (Von Halle, 2001). Her approach is a combination of various ideas, which often leave plenty of space for improvisation for analysts and designers. Some of those ideas indeed cannot be overlooked, e.g., the use of business decisions to help identify rules that are important to a particular event. However, the approach lacks formal presentation and is oriented towards the design of BR-based systems. We, on the other hand, aimed to create the requirements specification, which would be equally useful during the development of IS of any architecture (where business rules are stored and processed separately from process and data related IS components *or* where rules are within those components). In other words, we did not try to link requirements to specific design stage results.

It must be noted that business rules can be successfully structured using formal rule models. Our research, results of which were presented in (Butleris *et al.*, 2002), showed that a repository can be build for rules, expressed using a highly detailed Ross method (Ross, 1997). However, the management of such rules is too complicated for the requirements specification stage. Generally we consider formal BR representation models (e.g., Ross, OCL) to be better suited for the design stage of the (preferably BR-based) IS (Kapocius *et al.*, 2005).

The strategic decisions and assumptions that were made after summarizing our research and formed the basis for our proposal were as follows (Kapocius *et al.*, 2005):

- During the requirements specification rules need to be expressed as explicitly as possible therefore the natural language templates-based BR model is used. Structural rules (namely terms and facts) should be captured as elements of a conceptual data model.
- Function Hierarchy Diagram creation is the basis of process analysis. In comparison, von Halle suggests the use of event, use-case, workflow or other forms of requirements expression (Von Halle, 2001). Note that at this stage we assume that only functions, which have to be computerized, are considered.
- The management of captured requirements is performed using the specialized requirements repository system. Static structure model of the repository is an integral part of the approach as it allows a firm and unambiguous association between structural rules, non-structural rules and business functions.

3. BR-Based Requirements Specification Method

A schematic representation of the proposed requirements specification process is given in Fig. 1. The framework of this process was presented in considerable detail in (Kapocius *et al.*, 2005), therefore here we will only overview the basic features of the method that evolved out of this framework.

In the beginning of the process a business context has to be defined and actors that can be the future users of the system or simply a source of information for the analyst have to be registered. Initial analysis of the universe of discourse is well defined in various sources (Barker *et. al.*, 1992; Hull *et al.*, 2002; Kruchten, 1998; Maciaszek, 2001;

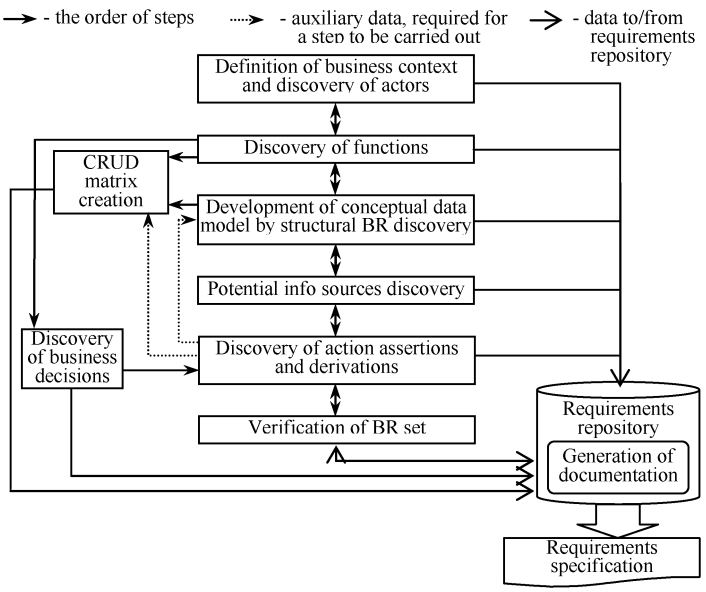

Fig. 1. The proposed process of the BR-based requirements specification method.

Robertson *et al.*, 1999) therefore only basic guidelines are outlined in our approach. Important feature of the proposed process is the inclusion of the creation of CRUD (data events: create, read, update, delete) matrix, the aim of which is to reveal how a given term (in this case: conceptual entity or attribute) is used during the performance of a given function (Von Halle, 2001). The discovery of non-structural business rules is performed on the basis of business decisions that need to be made during the course of the relevant function execution. Note that here business decisions are expressed in the form of questions that need to be answered in order to carry out the function.

As a schema of non-structural rule classification and expression natural language sentence template-based *Business Rules Solutions RuleSpeak* (BRS RuleSpeak) model (Ross, 2003; Ross *et al.*, 2001) was chosen. In RuleSpeak it is assumed that both terms and facts are already captured in a fact model (Ross *et al.*, 200a; 200b). This requirement was among the reasons why we chose to capture structural BR as elements of conceptual data model. Extended Entity Relationship model (EER) (Elmasri *et al.*, 2002) was chosen as a preferable standard because it includes ISA relationship and is fairly similar to the fact model. Of course, a natural language templates-based model could be developed for terms and facts, but it would have to be sophisticated enough to make conceptual data model generation possible. Therefore it would consist of ten or so structural BR types, some of which would require complicated templates to be used (Butleris *et al.*, 2002).

It appears BRS RuleSpeak has not been used in any known requirements specification methodologies because the authors of such methodologies (e.g., MBRM (Kardasis *et al.*, 2004; Wan-Kadir *et al.*, 2003; 2005), von Halle method (Von Halle, 2001), T. Morgan approach (Morgan, 2002)) tried to create their own templates. However, we think that

RuleSpeak is sophisticated enough for the task of BR expression and the purpose of a new classification is doubtful.

In the following sections of this paper we will discuss the backbone of the implementation of the proposed method – the static structure model of the requirements repository. Implementation of the repository will also be discussed.

4. The Architecture of the Requirements Repository

IS requirements discovered using the proposed method should be processed using a specialised Requirements repository system, at the core of which is the requirements database (repository). Repository IS can be viewed as consisting of five major subsystems:

1. Data sources subsystem (see Fig. 2 for static structure model);
2. Functions and business decisions subsystem (see Fig. 2);
3. Conceptual data model (structural BR) subsystem (see Fig. 2);
4. Non-structural BR templates subsystem (see Fig. 3);
5. Non-structural BR subsystem (see Fig. 3).

Note that in Figs. 2, 3 the bolded attribute is mandatory, while the underlined attribute is a primary key (no additional meaning). In the following sections of this section we will discuss the static structure of all mentioned subsystems.

4.1. *Data Sources Subsystem*

It is assumed that the sources of information during the requirements discovery can be people, documents or the program code of the existing system. All sources should be classified and stored in a requirements repository. There has to be a possibility to relate documents and program code elements to people, who have provided this information or can be helpful during the analysis of this information. Requirements repository must support the storage of exhaustive information about executive as well as regular staff members, who could be helpful during the requirements discovery process and/or are potential future users of the system under development. It is essential that all executive or management positions (or organization's structural units, if those units have a lot of employees, whose positions are not important) and relationships between them can be stored. Therefore table *Person* is intended for the storage of personal details of organization members or employees while table *Incumbency* stores information about administrative hierarchy of organization (see Fig. 2). Note that every captured non-structural BR must be related to one or more of its sources (table *BR_source*).

4.2. *Functions and Business Decisions Subsystem*

Results of the function discovery are Function Hierarchy diagrams. We assume that *only* functions that need to be computerized are captured (Kapocius *et al.*, 2005).

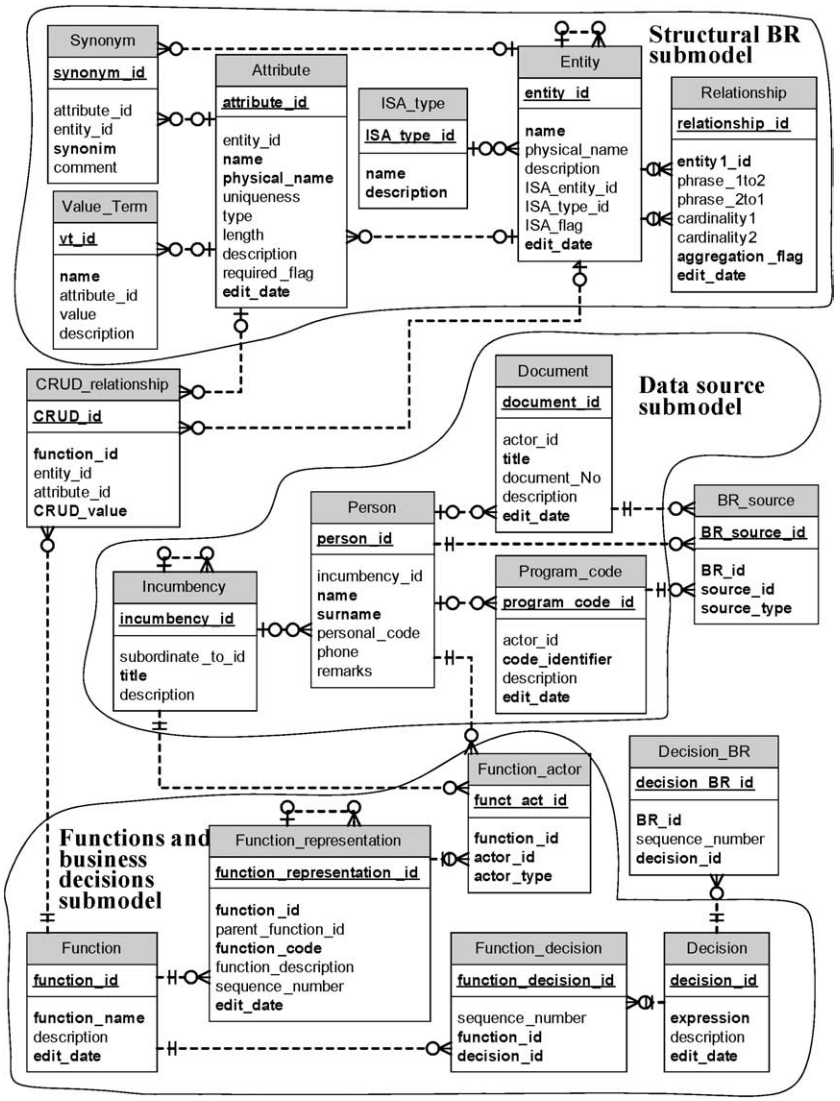

Fig. 2. Fragment of a conceptual data model of the proposed requirements repository.

An important feature of our method is the possibility to assign sequence numbers to sub-functions within a parent-function (attribute *Function_representation.sequence_number*). This is required in order to capture the sequence of the parent-function's execution (it will have direct influence on underlying business rules' execution sequences as well). It is also important to note that each repeat (or clone) of a function is treated as a separate representation of this function (table *Function_representation*). Function representations can be related to one or more actors (via table *Function_actor*), each of which can be either a person or an incumbency (attribute *actor_type*). This is important in order to simplify upcoming analysis tasks.

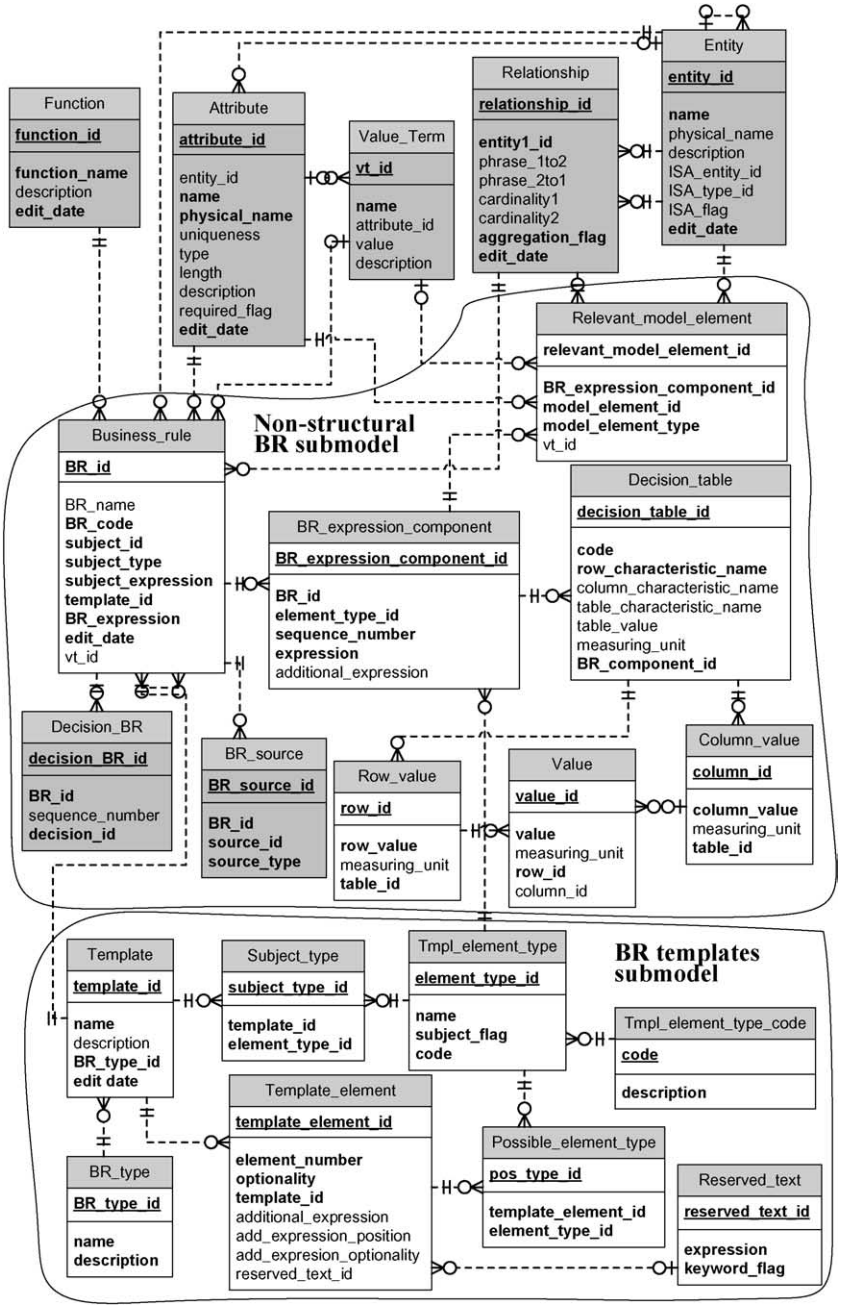

Fig. 3. Fragment of a conceptual data model of the proposed requirements repository (darker tables were already displayed in Fig. 2).

Tables' *Function* and *Function_representation* attribute *edit_date* indicates when a particular function or function representation was created or when the last change to any of its attributes was carried out. If a row from any of the mentioned tables has to be deleted, the value of *edit_date* could be set to a predefined value (e.g., 1111.11.11) with the rest of the information left unchanged. This way the management of requirements model's elements, which are related with a "deleted" data, can be simplified. In this case such related elements are decisions and business rules that were discovered during the analysis of the deleted function. Similar attributes are also used in other tables of the model, as it will be shown in the upcoming sections of this section.

Business decisions, which are expressed in a form of questions that need to be answered in order to perform a given function (Von Halle, 2001), have to be related to functions. In order to carry out a function more than one decision may be required. Similarly, each decision may be important for several functions (Kapocius *et al.*, 2005) (hence table *Function_decision* is used). Note that all representations (or repeats) of the same function will be related to the same group of decisions. Sometimes, in order to carry out a function, decisions have to be taken in a certain order (attribute *Function_decision.sequence_number*).

Let's have a look at an example. We will use a simple scenario of car insurance activity. One of the functions is "estimate the value of the car". Actor of this function is an unspecified person (incumbency) "insurance agent". In this case captured business decisions could be as follows: "Does the car qualify for estimation?", "What is the value of the car?". We will use this example in the upcoming sections of this section.

4.3. *Conceptual Data Model (Structural BR) Subsystem*

According to the proposed requirements specification method, structural BR will be captured as conceptual data model elements – entities, attributes and relationships between entities. Note that "Is a" (ISA) and aggregation relationships should also be recordable (hence the closed relationship on a table *Entity* Fig. 2). ISA relationship would be identified by ISA relationship type stored in table *ISA_type* (relationship can be "type", "format", "nature", "sort", etc.), while aggregation relationship is indicated using attribute *Entity.aggregation_flag*. A textual expression of such elements can be generated using their descriptions. Each term (i.e., entity or attribute name) may have one or more synonyms (table *Synonym*, Fig. 2). The sub-model, which will be used in the upcoming examples, is presented in Fig. 4.

Apart from entities, attributes, relationships and synonyms the proposed model also supports the storage of attribute values and terms that cannot be included into a conceptual data model (table *ValueTerm*). During the discovery of requirements three types of such terms and values can be found:

1. The fixed values of attributes. For example, an attribute "colour" of the entity "Car" can have an important value "most popular colour". In this case in the model's table *ValueTerm* columns *attribute_id* and *value* must be filled. If value is calculated using rules, attribute value is empty.

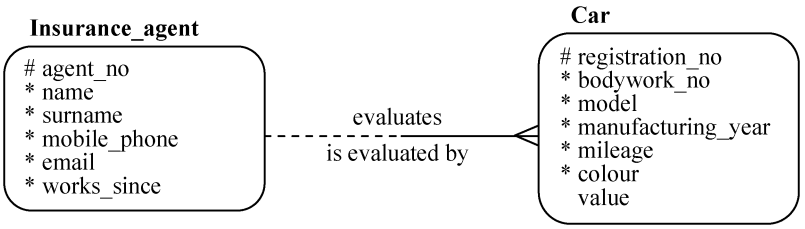

Fig. 4. Fragment of a conceptual data model of car insurance activity.

2. Fixed independent value (constant). Examples of such values include various coefficients, tax rates and so on. In case of such values table's *ValueTerm* column *attribute_id* will be empty while column *value* will be full.
3. Terms that cannot be included into a conceptual data model. Often specific terms, abbreviations or expressions that are used in the universe of discourse cannot be treated as elements of the conceptual data model and should be gathered in a glossary (e.g., GTI, internal corrosion). In case of such terms/expressions table's *ValueTerm* columns *attribute_id* and *value* will both be empty.

As it was already mentioned, terms should be related to functions via CRUD matrix. All CRUD relations are stored in a table *CRUD_relationship* in a form of relevant operation abbreviations, separated by comma. A simple example of the CRUD matrix for one function is given in Table 1. The table shows that in order to perform the evaluation of the car (i.e., create or update the attribute *Car.value*), its model, manufacturing year and mileage must be known (i.e., read from database).

Table 1
An example of the CRUD matrix for one function

Functions	Terms			
	car value	car manufacturing year	car model	car mileage
Estimate the value of the car	C, R, U	R	R	R

4.4. *Non-Structural BR Templates Subsystem*

According to BRS RuleSpeak, every rule besides its functional category also has an explicit subject at the beginning of rule statement. The subject of the rule can be a term, data item, fact, other rule, process or procedure. Each category of BR has its own keyword and structural template, which can be decomposed into individual parts and stored in a repository. Each stored template can be edited.

For example, a template for imprint rule type BR is (Ross, 2003):

<Subject> must/should [not] BE SET to <term/value> [when/if <condition>]

Here the subject can be either a term or a fact (Fig. 3, table *Subject_type*). Attribute *Tmpl_element_type.subject_flag* indicates whether a stored type can be assigned to subject. Aforementioned template would be stored as a sequence of four elements (subject not included):

1. "must/should [not]". This is a reserved text type element with an optional additional expression "not" (attribute *Template_element.additional_expression*) that goes before the element (attribute *Template_element.add_expression position*).
2. "BE SET to". This is a keyword – an expression, which is unique for imprint rule. Note that both keywords and other reserved expressions are stored in a table *Reserved_text*, but separated using attribute *keyword_flag*.
3. "<term/value>". This element can be either a term or a value (value set). Cases like these are the reason for the table *Possible_element_type*.
4. "[when/if <condition>]". This is a condition type element. It has an additional expression "when/if" (two choices), which is required and goes before the element.

Table *Tmpl_element_type_code* stores the predefined codes of element types that are required in order to identify particular elements when writing the program code for the requirements repository IS. These codes are the same independent of the language used (e.g., English, Lithuanian, etc.), while the names of element types stored in table *Tmpl_element_type* can differ. Some codes cover more than one element type because of the identical properties of those types (from the view of the repository IS implementation).

Note that attribute's *Template.edit_date* value will indicate date and time when any of template's elements changed (it will also show if the template has been deleted).

4.5. *Non-Structural BR Subsystem*

Every use of structural BR in a non-structural rule should be recorded in the requirements repository by associating relevant entities, attributes and relationships to the captured rule (see Fig. 3). However, the wordings of those elements in BR statements may differ.

According to the principles of BRS RuleSpeak, the subject of the rule can be an entity, an attribute, an attribute value, a relationship, a function or other rule (attribute *Business_rule.subject_type*). However, various terms and facts can be used in rule's expression components (table *BR_expression_component*), therefore such components (e.g., conditions, prepositions, mathematical formulas) should also be linked to the relevant values and elements of the conceptual data model (table *Relevant_model_element*). Note that the exact position of the referred facts, terms and values in the wording of rule's components is not recorded. Loose relation is sufficient in order to give analyst the ability to trace rules by the terms and facts used in them.

The attribute *Business_rule.edit_date* will show when any of rules components were last modified. This way it is possible to identify any rules that were last edited prior to the change of the underlying template. The identified rules can then be automatically or manually edited to comply with a new version of the template.

Links between decisions and rules are recorded in a table *Decision_BR*. Optional attribute *Decision_BR.Sequence_Number* would be used to capture the order of rule enforcement within a particular decision if required.

At the end of section 4.2 we had a simple function and two of its business decisions. The following rules were discovered analysing those business decisions: "R1: the car manufacturing year must be at least 1993 for the car to qualify for estimation", "R2: the car value must be set to the value in Table T for the given model, the year of manufacturing and mileage". Here rule R2 references a decision table (not to be confused with the business decisions described above), because the value of the car cannot be calculated using a formula. Decision tables would be stored in a simple data structure, consisting of four tables: *Decision_table*, *Row_value*, *Column_value* and *Value*. Note that each BR can refer to one or more decision tables, each of which can carry a single table value (attribute *Decision_table.table_value*) in addition to row and column values. Therefore the model supports decision tables involving from one to three evaluation terms or characteristics (the latter would be split into array of tables, see Table 2). Table 3 shows how one of the discovered rules (imprint rule type rule R2) would be stored in the requirements repository.

5. Implementation of the Requirements Repository System

Implementation of the presented requirements repository system should consist of five subsystems as divided in this paper. Graphical user interface should be used wherever possible. It is important to note, that the static structure of the repository was designed with automation in mind. In other words, information in the DB should be used to generate rule-editing forms (according to the templates structure), automatically update rules' statements when the relevant templates change, etc.

Hierarchical structures should be used for all subsystems as they allow visualizing complicated structural features without using graphical interface. Two windows from a prototype we are currently using are presented in Fig. 5. The figure shows both attribute and graphical interface windows for functions-decisions-BR hierarchy management. The

Table 2

An example of a decision table array (a small fragment)

Car model	Table T(1). Mileage $<$ 100000 km				T(2). Mileage $\geqslant$ 100000 km			
Year	1994	1995	1996	1997	1994	1995	1996	1997
Mitsubishi Lancer 1500 GLXi KC	too old	2050	2360	2800	too old	2000	2300	2700
Mitsubishi Lancer 1800 GTi-16v KC	2000	2800	3300	4000	1950	2750	3250	3900

Table 3
An example of BR decomposition for storage in a repository

Data	Stored in table(-s)	Related to table	Remarks
"The car value must be set to the value in Table T for the given model, the year of manufacturing and car mileage"	*Business_rule*	*Entity* via attribute *subject_id*	The rule's subject "car value" is an attribute *value* of the entity *Car* in the conceptual data model (Fig 4.)
		Template via attribute *template_id*	The rule uses a template "must/ should [not] BE SET to"
"must"	*BR_expression _component*	*Tmpl_element_type* via attribute *element_ type_id*	In this case "must" is a *predefined text*
		Business_rule via attribute *br_id*	–
"be set to"	*BR_expression _component*	*Tmpl_element_type* via attribute *element_type_id*	"be set to" is a *keyword*
		Business_rule via attribute *br_id*	–
"the value in Table T for the given model, the year of manufacturing and mileage"	*BR_expression _component*	*Tmpl_element_type* via attribute e*lement_type_id*	In this case this component is a *value set*
		Attribute via table *Relevant_model_ element*	This component refers to attributes of the entity *Car* (Fig.4): *model*,*mileage* and *manufacturing_year*

tool supports full synchronisation of data between the graphical and attribute interfaces (if they are being used simultaneously).

Various reports and diagrams are vital in order to produce a detailed and versatile requirements specification. Here are some documentation forms that the system should be able to generate:

- function Hierarchy diagrams by hierarchy level or by high-level function;
- functions-decisions-rules hierarchy diagram and its variations;
- functions-rules tables for any function or a set of functions;
- business decisions-rules tables for any decision or a set of decisions by function;
- the model of the administrative structure of the organization;
- the conceptual data model – the Extended Entity-Relationship (EER) diagram;
- glossary of terms;
- decision tables.

Conclusions

The analysis of current IS requirements elicitation techniques shows that they tend to be too much analyst-oriented, while known BR modelling methods do not represent suf-

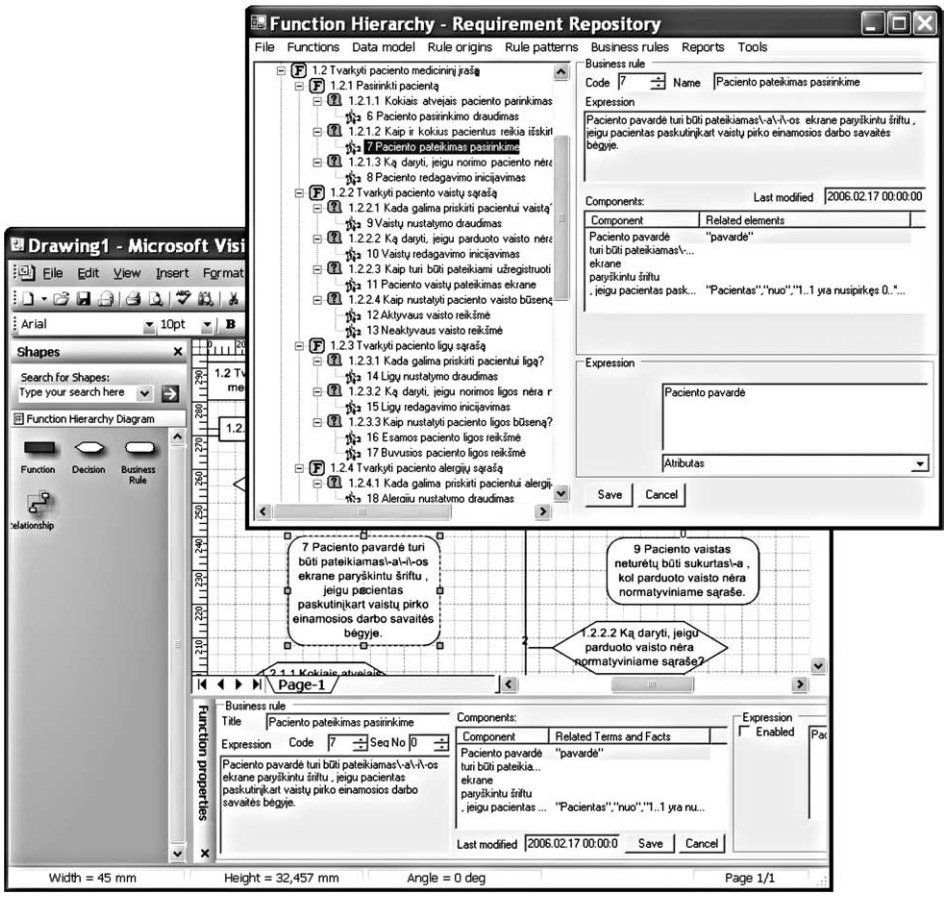

Fig. 5. Requirements repository prototype's functions-decisions-rules hierarchy management tools.

ficiently the natural order of universe of discourse analysis. These shortcomings are addressed in the proposed framework for the BR-based requirement elicitation process. The framework was the basis for a new requirements specification method, which is centred on the business rules discovery through the analysis of functions and underlying business decisions. Structural business rules are captured in the form of conceptual data model elements while all of the remaining rules are expressed and structured using natural language templates proposed in BRS RuleSpeak (Ross, 2003; Ross *et al.*, 2001).

One of the features of the approach is the introduction of a single repository for the captured requirements. Using the static structure model of the repository, which was presented in this paper, a flexible requirements specification system can be created. The most important aspect of the presented repository is that it facilitates the modelling of requirements in a manner close to natural thus minimising the gap between analysts and stakeholders and improving the quality of specification. It could be applied during the development of any type of IS. Proposed requirements repository would also be use-

ful during the development of web-based systems, which rely heavily on business rules. Multi-layered architecture is often used in web applications, where multipurpose implementations of systems components can provide autonomous services to various clients.

The future work includes further studies of the proposed method using a fully functioning prototype of the requirements specification system. The application of BR-based requirements during the IS design phase is the major challenge that will be addressed.

References

Barker, R., and C. Longman (1992). *CASE*METHOD: Function and Process Modelling*, Addison-Wesley.

Butleris, R., and K. Kapocius (2002). The business rules repository for information systems design. In *Research Communications of 6th East European Conference ADBIS 2002*, Vydavatel'stvo STU, Bratislava. pp. 64–77.

Butleris, R., and K. Kapocius (2003). Structuring of business rules during the information system development. In *Proceedings of Pre-Conference Workshop of VLDB 2003 "Emerging Database Research in East Europe"*, Computer Science Reports, Report 14/03, BTU Cottbus, Cottbus. pp. 17–21.

Hall, J., K. Anderson Healy and R. G. Ross (Eds.) (2000). *Defining Business Rules: What Are They Really?* (3rd ed.), Business Rules Group.
`http://www.BusinesRulesGroup.org.`

Demuth, B., H. Hussmann and S. Loecher (2001). OCL as a specification language for business rules in database applications. *Lecture Notes on Computer Science*, **2185**, 104–117.

Elmasri, R., and S. B. Navathe (2002). *Fundamentals of Database Systems*, Third Edition, Addison–Wesley.

Hull, E., K. Jackson and J. Dick (2002). *Requirements Engineering*, Springer Verlag, London.

Kapocius, K., and R. Butleris (2005). Business rules driven approach for elicitation of IS requirements. In *Proceedings of the 9th World Multi-Conference on Systemics, Cybernetics and Informatics (WMSCI 2005)*, Vol. IV, Orlando. pp. 276–281.

Kardasis, P., and P. Loucopoulos (2004). Expressing and organising business rules. *Information and Software Technology*, **46** (11), 701–718.

Kruchten, P. (1998). *The Rational Unified Process* (An Introduction), Addison–Wesley–Longman, Inc.

Leonardi, M.C., and J.C.S.P. Leite (2002). Using business rules in extreme requirements. *Lecture Notes in Computer Science*, **2348**. In *Proceedings of 14th International Conference CAiSE 2002*. pp. 410–436.

Maciaszek, L.A. (2001). *Requirements Analysis and System Design: Developing Information Systems with UML*, Pearson Education Limited, Harlow.

Morgan, T. (2002). *Business Rules and Information Systems*, Addison–Wesley.

Robertson, S., and J. Robertson (1999). *Mastering the Requirements Process*, Addison–Wesley, New York.

Ross, R.G. (1997). *The Business Rule Book*, 2nd ed., Business Rule Solutions, Houston.

Ross, R.G. (2000a). *What Are Fact Models and Why Do You Need Them?*, Part 1, BRCommunity.com.
`http://www.brcommunity.com/cgi-bin/x.pl/commentary/b008a.html`

Ross, R.G. (2000b). *What are Fact Models and why do You Need Them?*, Part 2, BRCommunity.com.
`http://www.brcommunity.com/cgi-bin/x.pl/commentary/b008b.html`

Ross, R.G. (2003). *Principles of the Business Rules Approach*, Addison–Wesley.

Ross, R.G., and G.S.W. Lam (2001). *The Do's and Don'ts of Expressing Business Rules*, Business Rule Solutions.
`http://www.brsolutions.com/rulespeak_download.shtml`

Taveter, K., and G. Wagner (2001). Agent-oriented enterprise modeling based on business rules. *Lecture Notes on Computer Science*, **2224**, 527–540.

TEMPORA Consortium (1993). *TEMPORA Methodology Manual*.

Von Halle, B. (2001). *Business Rules Applied: Building Better Systems Using the Business Rules Approach*, John Wiley & Sons, New York.

Wan-Kadir, W.M.N., and P. Loucopoulos (2003). Relating evolving business rules to software design. In *Proceedings of International Conference on Software Engineering Research and Practice (SERP 2003)*, Las Vegas. pp. 129–134.

Wan-Kadir, W.M.N., and P. Loucopoulos (2005). Linking and propagating business rule changes to IS design. In *Information Systems Development – Advances in Theory, Practice and Education. Proceedings of ISD'2004 Conference*. Springer Science + Business Media, New York. pp. 253–264.

K. Kapočius was born in 1977, Kaunas, Lithuania. He received BCs and MCs degrees in informatics from Kaunas University of Technology, Lithuania in 1999 an 2001 respectively. Currently he is a younger scientific officer and a lecturer at the Department of Information Systems of Kaunas University of Technology and is pursuing the PhD degree at the same university. K. Kapocius is the author or coauthor of 15 scientific articles. His scientific interests include user requirements specification and information systems design using business rules approach, business rules structuring models and rules-driven IS implementation techniques.

R. Butleris is a professor at the Department of Information Systems of Kaunas University of Technology, member of ECCAI and Computer Society of Lithuania. In 1980 he graduated from Kaunas Institute of Polytechnics (now Kaunas University of Technology). In 1988 he received a PhD in control in technical systems from Kaunas Institute of Polytechnics. Currently his research interests include requirements engineering, business rules specification, information system design and reengineering.

Veiklos taisyklėmis grindžiamų IS reikalavimų saugykla

Kęstutis KAPOČIUS, Rimantas BUTLERIS

Programinės įrangos projektų kokybė neretai nukenčia dėl pernelyg didelės takoskyros tarp pavidalo, kuriuo reikalavimus pateikia užsakovas, ir formos, kuria šie reikalavimai užrašomi analitiko sudaromoje specifikacijoje. Siekiant spręsti šią problemą, buvo sukurtas naujas informacijos sistemų (IS) reikalavimų specifikavimo metodas. Šiame straipsnyje pristatoma viena svarbiausių metodo dalių – specifikuojamų reikalavimų saugyklos architektūra. Pagrindiniai reikalavimų objektai, kurių saugojimą ir apdorojimą palaiko sukurtasis saugyklos modelis, yra funkcijos, veiklos sprendimai, duomenų šaltiniai, koncepcinio duomenų modelio elementai, veiklos taisyklės ir jų užrašymo šablonai. Straipsnyje taip pat apžvelgiami svarbūs pristatomos saugyklos realizavimo aspektai.