Liet. matem. rink, **45***, spec. nr.*, 2005, 174–179

The use of formal languages for business rules realisation in information systems

Olegas VASILECAS, Evaldas LEBEDYS (VGTU)
e-mail: olegas@fm.vtu.lt, evaldas@isl.vtu.lt

1. Introduction

Business rules approach became the object of many researches in recent years. Rules are important components of every system. The whole of business rules make a policy of particular system. A policy describes the conditions under which actions are permitted or forbidden [2]. There are many different business rules in each business system. Business rules are classified differently for different purposes [11]. Business rules of different classes are represented differently in business system model. There are many modelling languages suitable to represent some set of business rules but the dominant opinion is that there is no single modelling language suitable to model all business rules. Even if some modelling language can be used to represent all business rules, the created business rules model may become hard to understand [12]. The need of different representations of business rules is also widely discussed in the papers of recent years. Business audience prefer informal business rules specification [8]. Formal business rules specification is needed for automated business rules implementation in information systems [1]. It is difficult to create abstract, informal business rules specification suitable for automated implementation. We propose to create formal business rules specification using information about business rules represented in UML diagrams. Formal business rules specification is proposed to be created using first order logic constructions. The way business rules of different classes are expressed formally is presented in the paper.

The rest of the paper is organised as follows. The second section shortly discusses business rules classification and presents the classes of business rules analysed further in the paper. Section 3 briefly analysis the use of Unified Modelling Language for representation of business rules of different classes. Section 4 shortly describes the elements of first order logic used to create formal business rules statements. Section 5 presents the way business rules are expressed formally using first order logic. Section 6 concludes the paper.

2. Business rules classification

Business rules can be classified using different taxonomies [6]. Business rules classification depends on the point of view business rules are classified. Business rules can be

classified according to the way business rules are implemented in information systems [5]. Different classes of business rules can be distinguished while analysing modelling languages most suitable for specific kinds of business rules. Business rules may also be classified according to the structure of rules statements. We distinguish classes of business rules according to the way business rules influence business. There are three main classes of business rules:

- business rules describing the roles and obligations of system actors;
- business rules specifying and constraining the structure of the systems objects and the relationships between these objects;
- business rules describing the behaviour of business system objects and actors.

The representation of business rules of these classes is discussed further.

3. Business rules representation

There are many modelling languages, having both graphical and textual notations, suitable to model business rules of different classes: Unified modelling language (UML), IDEF, entity relationship diagrams, conceptual graphs, decision trees, decision tables, first order logic and so on [11]. Formal methods for specifying design patterns, extensions of existing modelling languages and new business rules modelling languages are proposed [3], [4]. But there is no single modelling language suitable to model all business rules. UML has become the most widely used modelling language for business systems and information systems modelling in recent years [9]. Business rules appear in different UML diagrams. Not all business rules can be represented in UML diagrams. Some complex logic describing business rules are hardly represented using graphical notations. Formal methods, such as first order logic can be used to represent rules describing complex logic [12].

Business rules of different classes are represented in different UML diagrams (Fig. 1):

- business rules expressing the roles and obligations of system actors are represented in Use Case diagrams;
- business rules specifying and constraining the structure of the systems objects and the relationships between these objects are represented in UML classes and objects diagrams;
- business rules specifying the behaviour of business system objects and actors are represented in sequence, collaboration, activity and state transition diagrams. Sequence and collaboration diagrams contain the same business rules and are not analysed separately in this paper.

Use Case diagrams are used to present the tasks assigned to different actors of the system of interest. Business rules in Use Case diagrams appear as statements describing system actors obligations.

Class diagrams are used to represent classes of objects. Classes are aggregates of objects which are used as variables in predicates when specifying business rules formally. In other words, classes are the sets of business objects.

Sequence and collaboration diagrams are used to represent the sequence of actions and operations needed to perform particular task. State transition diagrams are used to

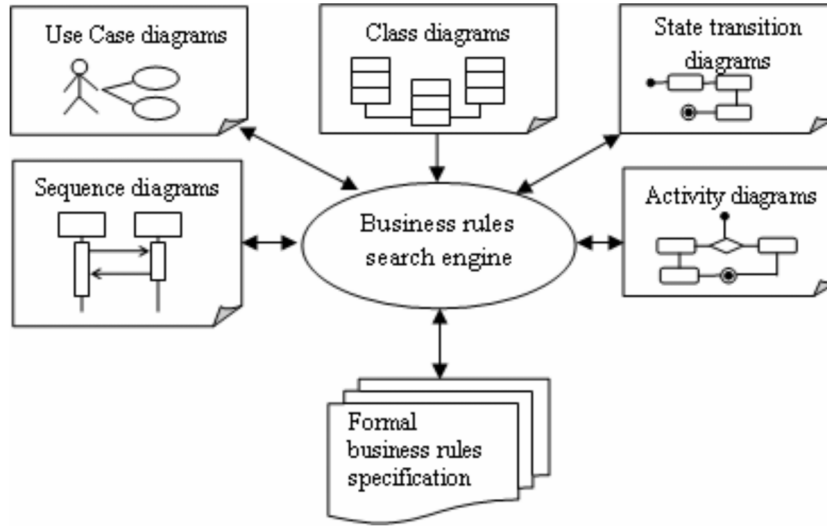

Fig. 1. Specification of business rules resented in UML diagrams.

specify the sequences of changes of states of business objects. Activity diagrams are used to model the logic of the operations captured by a use case or a few use cases. Activity diagrams can be used to represent both the basic sequence of actions as well as the alternate sequence of actions.

UML contains other diagrams: package, component and deployment diagrams. These diagrams are not discussed in this paper as they are used to represent aspects of the development and implementation of software systems and the purpose of this paper is to discuss rules appearing in business system models.

4. First order logic

The language of first order logic is built on objects and relations. Facts about the objects can also be expressed using first order logic. Generally, first order logic is used to represent facts, objects and relationships [10]. We use nouns in natural language which refer to objects in first order logic. For example, a manager, a client, an invoice, a car and so on. Verbs in the natural language refer to relations among objects. For example, is employed, has credit, is red and so on. Some of these relations are functions. Functions are the relations which have only one value for the particular input. Relations can be unary, binary and polynomial. A sentence is a formula which has no free variables [7].

The basic syntactic elements of first order logic are the symbols used to specify objects, relations and functions. These symbols are [10]:

- constant symbols – are used to represent objects;
- predicate symbols – are used to represent relations;
- function symbols – are used to represent functions.

The statements in first order logic are expressed in sentences. Sentences may be:

- atomic sentence;
- complex sentence – a sentence composed of a two or more atomic sentences;
- quantifier Variable,... Sentence;
- negation of a sentence.

Atomic sentences are formed as:

- predicate(Term, ...);
- Term=Term.

Term is one of the: Function, Constant, Variable.

Complex sentences are formed from atomic sentences using connectives. A few connectives are used in first order logic uses: "If ... Then ...", "And" "Or", "If and only If". Quantifiers are used to form sentences and are "For all" and "Exists". Constants specify objects. Variables are the elements which value may change and it impacts the output. Predicates are used to specify relations and generally may become true or false depending on the particular situation. Functions are used to represent relations which may return any value.

5. Business rules formal representation

Business rules are expressed using first order logic as sentences, both atomic and complex. All basic syntactic elements of first order logic can be used to express business rules. Only business rules of classes specified above are analysed in this paper. Business rules of various classes have different structure. Business rules of different classes are represented formally using different syntactic templates [12]. This means that all business rules of particular class can be specified using some set of syntactic templates. Templates for expressing formally business rules represented in UML diagrams are presented further.

Business rules represented in Use Case diagrams can be expressed formally depending on the way tasks are assigned to actors. For example, a fragment of Use Case diagram can be interpreted in either of two ways "Bookkeeper has to register the invoice" or "Bookkeeper may register the invoice". The role has to be declared, to express business rules. Business rules appearing in Use Case diagrams can be expressed using predicate P1 = Has_to(A,T) and is read as "Actor A has to perform task T".

Class model includes business rules representing constraints of business objects, properties of relationships between business objects. Rules describing the properties of relationships between classes can be described using predicate: P2 = Related(C1,C2) where predicate is "Each object of class C1 must be related to some object of class C2". The rules describing allowable ranges of values for objects properties can be described using predicate P3 = (o∈C) o.property <,>,=,=/ P, where "o" is an object, "C" is class of objects, "o.property" is concrete property of an object, "P" is parameter (constant value).

Event-Condition-Action (ECA) rules are mostly represented in state transition diagrams. Such a rule can be expressed formally using a predicate of a special form: P3 = (E, C, A1, A2) where predicate is "On event E, if condition C, then action A1, else action A2". This kind of predicate is used to represent ECA rules.

Activity diagrams mostly contain ECA business rules. Rules represented in activity diagrams are represented formally in the same way as business rules represented in state transition diagrams. Mostly business rules represented in activity diagrams can be expressed formally using predicate P4 = (A1,C1,A2) where predicate is "When action A1 is performed, if condition C1 is/is not satisfied, then action A2 is performed.

Generally, sequence and collaboration diagrams include business rules describing the exact order of actions for a user to be taken to perform specific task. A predicate of a special form is used for describing rules specified in sequence diagrams. For example, predicate "An action A2 is performed after actions A1 is done" is expressed formally as P5 = (A1, A2).

6. Conclusions

Business rules approach is important because business rules influence almost all aspects of business systems. Literature analysis showed that business rules can be classified differently depending on the purpose business rules are analysed with. Business rules of different classes can be represented using different modelling languages. The choice of the modelling language depends on the purpose of the model. We proposed to create formal business rules specification using first order logic constructions. We also presented syntactic templates used to express business rules represented in UML diagrams formally. First order logic elements were used to create syntactic templates. The tool implementing proposed above is under development.

References

1. D. Bevington. ASL – a formal language for specifying a complete logical system model (Zachman row 3) including business rules, *Business Rules Journal*, **5** (1) (2004). `http://www.BRCommunity.com/a2004/b167.html`. Retrieved on 2004.05.10 13:21.
2. J.Y. Halpern, V. Weissman, Using first-order logic to reason about policies, in: *Proceedings of 16th IEEE Computer Security Foundations Workshop (CSFW'03)*, IEEE Computer Society Press (2003), pp. 187–201.
3. C.E. Hughes, A. Orooji, D.R.E. Williams, A mathematical formalism for specifying design patterns, in: *Proceedings of 17th International Symposium on Computer and Information Sciences (ISCIS XVII)*, Orlando, FL, Oct. 28–30 (2002).
4. A. Kovacic, The rule transformation approach to business renovation, *Business Rules Journal*, **4** (8) (2003). `http://www.BRCommunity.com/a2003/b162.html`. Retrieved on 2005.04.28.
5. J. Laucius, E. Lebedys, O. Vasilecas, Realisation of event-condition-action rules using active database system triggers, *Information Sciences*, Vilnius University, 129–133 (2003) (in Lithuanian).
6. J.C.S.P. Leite, Ma. Carmen Leonardi, Business rules as organizational policies, in: *IEEE IWSSD9: Ninth International Workshop on Software Specification and Design*, IEEE Computer Society Press (1998), pp. 68–76.
7. D. Marker, *An Introduction to Model Theory*, Springer-Verlag (2001).
8. T. Morgan, *Business Rules and Information Systems: Aligning IT with Business Goals*, Addison-Wesley Pub Co (2002).
9. G. Sparks, *An Introduction to Modelling Software Systems Using the Unified Modelling Language: The Dynamic Model*. `http://www.sparxsystems.com.au/WhitePapers/The_Dynamic_Model.pdf`. Retrieved on 2005.04.20.

10. J.R. Stuart, P. Norvig, *Artificial Intelligence: A Modern Approach*, Prentice Hall, Upper Saddle River, New Jersey, second edition (2003).
11. O. Vasilecas, E. Lebedys, Formal business rules specification for automation of engineering process. To be published in the *Proceedings of the 19th International Conference "Systems for Automation of Engineering and Research" (SAER'2005)*, Varna, Bulgaria (2005).
12. O. Vasilecas, E. Lebedys, J. Laucius, Formal methods for representation of business rules specified using UML, in: R. Simutis (Ed.), *Proceedings of the International Conference "Information Technologies for business – 2005"*, Kaunas, Technologija, pp. 41–47.

REZIUMĖ

O. Vasilecas, E. Lebedys. Formalių kalbų panaudojimas verslo taisyklių realizacijai informacinėse sistemose

Paskutiniu metu vykdoma daug tyrimų informacinių sistemų modeliavimo verslo taisyklių požiūriu kryptimi. Sistemų modeliavimas taisyklių požiūriu naudingas, siekiant automatizuoti programų sistemų kūrimo procesą. Literatūroje vyrauja nuomonė, kad nėra vienos modeliavimo kalbos, tinkančios sukurti išsamų koncepcinį dalykinės srities modelį ir aprašyti visų klasių verslo taisykles. Norint automatizuoti programų sistemų kūrimo procesą, reikalinga formali verslo taisyklių specifikacija, tinkama tolimesniam automatizuotam apdorojimui. Darbe siūlome informacijos apdorojimo taisykles užrašyti naudojant pirmos eilės predikatų logikos sakinius. Darbe pavaizduota, kaip skirtingose UML diagramose pavaizduotos verslo taisyklės gali būti išreikštos formaliai, tai yra pateikti sintaksiniai šablonai suformuoti naudojant pirmos eilės predikatų logikos konstrukcijas.