UDC:004.8
Original scientific paper

ACTIVE DATABASES, BUSINESS RULES AND REACTIVE AGENTS – WHAT IS THE CONNECTION?

Kornelije Rabuzin, Mirko Maleković, Miroslav Bača
University of Zagreb
Faculty of Organization and Informatics, Varaždin
{kornelije.rabuzin, mirko.malekovic, miroslav.baca}@foi.hr

Abstract: *These three technologies were and still are mainly treated separately. Since not much work has been carried out in defining and combining them together, we are going to present what has been done and put accent on what could be done. Namely, they rely upon similar paradigms and concepts, as will be shown later on, and can be treated as complementary technologies. In this paper we will show that reactive agents react according to some set of business rules and active databases can be used as a suitable means for implementing business rules and in those way reactive agents as well. Since reactive agents have been well defined, recent improvements in the fields of active databases technology and especially business rules provide the reason to consider the benefits to be achieved from combining these fields.*

Keywords: *business rules, ADBMS, active database, agents, reactive agents*

1. INTRODUCTION

Although the relational model has been used for over 30 years, the development and use of new technologies, object oriented programming, real time systems etc. has resulted in emergence of different kinds of database systems, among which are also active database management systems (ADBMSs). ADBMS is a conventional database system capable of reacting to some events of interest which can occur within the database, or outside it. To understand how this works in practice, the basic concept on which an ADBMS relies – the concept of ECA, or active rules (ECA stands for Event-Condition-Action) – needs to be considered. According to it, when certain events occur (ON EVENT), and some conditions are fulfilled (IF CONDITION), some actions are performed automatically (THEN ACTION). These actions are performed without any need for the user's intervention. At the conceptual level people often talk about ECA rules; these rules are mostly implemented using triggers in some concrete ADBMS (more on ECA rules can be found in [0], [0] and [0]).

Another topic to be considered is business rules, which make a very important component of a business system. Unlike the generally accepted definition of ADBMSs, there is far less agreement on this topic. As a result, there are lots of different approaches and definitions of this term, which make it hard to determine what business rule really is.

Some of those definitions are going to be presented in the chapter about business rules, but at this point it needs to be said that they are a very important part of a business system because they influence the way business is run. Business rules can be found in many different forms; they can be written down in a manual, or implemented in form of computer programs, i.e. applications, used for running business. What is important is that business rules can be changed very often due to the complex environment in which a business system operates.

The last thing which has left for the introduction part is the term "agent". We have to say that this term was a buzz-word for a long time; but nowadays this term has been defined and many MultiAgent Systems (MASs) have been built and put into operation. We can say that systems which are capable of making decisions about actions they are to perform are treated as agents. Due to their properties like autonomy, mobility, reactivity and some abilities like communication, coordination and cooperation, agents are able to solve some complex problems and they can exhibit very complex behaviour; that is why there are many fields where agents are used (they can buy and sell some goods, find some information and so on). Although several different kinds of agent have been identified, we have accepted the classification according to which deliberative and reactive agents exist. These two types of agents will be explained later, but for now we will just say that reactive agents are very important because reactive behaviour is very often the best solution and a necessity in certain cases.

Since active databases and reactive agents rely upon the concept of reactivity, and taking into account that active databases have certain advantages compared to programs written in a programming language, we think that the concept of reactivity and these advantages make them suitable for reactive agent's implementation. On the other side, reactive agents have to obey some business rules when achieving a goal. So we have come to a question whether business rules – and what types of them – can be written within the active database.

One has to have in mind that business rules technology is declarative technology as well as active databases technology and writing a lot of programming code is a procedural approach; since both technologies are declarative, therefore we think it is suitable to use active database in order to implement a reactive agent and indirectly some set of business rules which this agent obeys. Of course, environment may be and usually is very dynamic and there is a possibility that business rules which determine the behaviour of some agent have to be changed very often; so it is easier to change something where you state what has to be done instead of something where you state how to do it. So the assumption about writing different kinds of business rules in active databases can be fully established.

Inasmuch as the term 'business rules' has been redefined, determining whether business rules – and what types of them – can be written in form of triggers, as well as analysing problems which may arise in the process when implementing reactive agents, presents a new challenge. The rest of this paper is organized as follows: Section 2 deals with ADBMSs, Section 3 describes business rules, Section 4 deals with agents (especially reactive agents), Section 5 discusses the combination of these technologies and Section 6 provides the conclusion.

2. ACTIVE DATABASE MANAGEMENT SYSTEMS

It has already been mentioned that ADBMSs have the capability of reacting automatically to certain events, which can occur within a database or outside it. An event can be defined as a state change of interest which requires intervention. From the point of view of ADBMSs these events of interest can be divided into two categories: simple and complex events. Simple events are the basic database operations like INSERT, UPDATE or

DELETE, or time events, which can be divided into absolute, periodic and relative events. Transaction events (for example, the beginning or the end of a transaction), method events (used in active object-oriented DBMSs) and abstract events are also treated as simple events. Complex events consist of one or more simple events connected with logical operators, but there are also special kinds of complex events like REPEAT, SEQUENCE or NEGATION. For example, if you have simple events E1 and E2, then $E1 \wedge E2$ or $E1 \vee E2$ represents a complex event. There are lots of techniques of discovering events. The more different kinds of events can be recognized, the better. More on different kinds of events can be found in [0], [0] or [0]. Generally speaking, event requires proper reaction. This reaction can be trivial, but mostly it is not. Since events can indicate the fact that certain business rules are broken, corrective actions need to be performed. Simple and complex events are very useful when trying to implement a reactive agent.

The event part of the active rule determines when the rule should be considered, the condition part determines whether the action part of the rule should be executed, and the action part of the rule represents the actions to be executed. The active rule is triggered when the event specified in the event part of that rule occurs. The triggered rule does not have to be executed; this depends on condition evaluation. Each ADBMS has a language, which is used for trigger specification (definition), and has an execution model, which determines how the rules are going to be executed. For example, Rock and Roll system's rule syntax can be found in [0], while PostgreSQL trigger syntax presented in [0] is:

```
CREATE TRIGGER name { BEFORE | AFTER } {event [OR ...]}
ON table FOR EACH { ROW | STATEMENT }
EXECUTE PROCEDURE func ( arguments )
```

Active databases are used in a lot of different areas, as can be found in [0]. They are used for performing simple tasks (for example, automatic reordering) as well as some rather complex ones (for example, in aircrafts, medical applications, etc.). Due to the increasing awareness about what active rules can do, papers have recently been published presenting how active rules can be used for maintaining XML files or in combination with OLAP systems, as can be found in [0] and [0], respectively, or in workflow management, as can be found in [0].

There are several arguments justifying the use of ADBMSs. First of all, it is cheaper to build such an application and its performances are better, at least when a small number of triggers is involved. Secondly, such an application is smaller and easier to maintain. Thirdly, they are an instance of declarative approach and, according to [0], "*the trend has clearly always been away from procedural and toward declarative – that is, from how to what*".

An ADBMS relies on a passive DBMS. Therefore, while building an ADBMS, a normal database system has to be extended in order to support active functionality; different kinds of events have to be detected, transaction management has to be improved because of different models of active rules execution, etc. In order to support this functionality, a passive DBMS can be extended using integrated, layered or application oriented approach as presented in [0] and [0].

When it comes to ADBMS performance, the awareness of what an ADBMS can do is just as important as the possibility to measure that performance. Perhaps some other active system could yield better results, or some bottlenecks could be discovered; that is why some tools have been introduced. One of the first tools used for performance measurement was Beast in the SAMOS project; later used was Objective. More on these tools can be found in [0] and [0], respectively.

As it has been already mentioned, when an event occurs, the condition is evaluated and then some actions are executed, provided that condition evaluation was successful. However, it is sometimes useful to postpone the condition evaluation or action execution so that they are not performed immediately, which explains why several different rules execution models exist. Thus the condition does not have to be evaluated or the action executed immediately after the event has been detected and the condition evaluated, but some time can pass in between. As a result, the condition can be evaluated at the end of the triggering transaction or the action can be executed in a new transaction, which does not depend on the triggering one. More on different execution models can be found in [0] and [0]. Different execution models are implemented using nested transactions (i.e. triggering and triggered transactions). More on transactions can be found in [0]. A technique for modelling applications in active object oriented DBMS has been presented in [0].

Another important question concerning ADBMS is the static analysis. Namely, it has been discovered that triggers sometimes do not exhibit the desired behaviour and the system behaviour is generally not predictable. It is possible that triggers trigger one another, so that the result is not always as expected. Redundant rules can also exist, and the sequence of rule firing is not always predictable. Therefore several approaches (triggering graph, activating graph, meta-rule analysis, etc.) for rule termination analysis and redundant rules check have been introduced, as can be found in [0], [0] and [0]. More about ADBMSs can be found in [0] and [0].

3. BUSINESS RULES

Although business rules have been gaining much attention lately, there is still no general agreement about what business rule really is, which explains why so many different definitions of this term exist. The fact that these definitions are rather dissimilar makes it hard to explain what business rule really is. The differences among those definitions may arise from different points of view, as will be shown later. Some of the definitions are listed below:

According to [0], business rules are defined as "*the set of conditions that govern a business event so that it occurs in a way that is acceptable to the business (or customer).*"

According to the Business Rule Group (BRG) in [0], a business rule is "*a statement that defines or constrains some aspect of the business. It is intended to assert business structure or to control or influence the behaviour of the business.*"

Ross has defined business rule in [0] as "*a directive intended to influence or guide business behaviour*".

According to [0], business rules are "*... the data rules which cannot be easily represented within a database.*"

Since many different definitions of this term exist, there are also many different classifications available. According to Date, as presented in [0], business rules can be classified into three main categories:

1. Presentation rules
2. Application rules
3. Database rules.

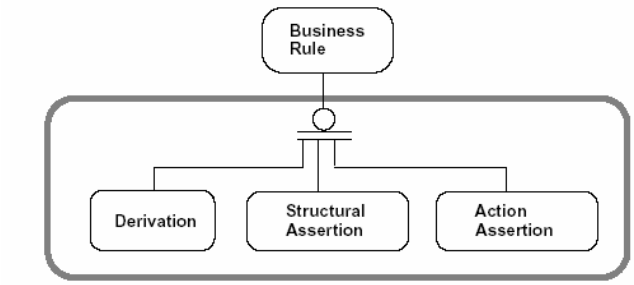
Presentation rules determine how data is going to be presented. Application and database rules are hard to distinguish. According to Date, "*it is hard to draw a sharp dividing line between database rules and application rules*". He has divided application and database rules into *constraints* and *derivations*. Constraints are divided into *state* constraints (defining the legal state or values of the database), *transition* constraints (defining legal changes from one state to another) and *stimulus/response* constraints (the combination of an event and action), while derivations are divided into *computations* (some formulae) and *inferences* (they infer additional facts).

Ross has presented the following classification of business rules in [0]:

1. Rejectors
2. Projectors
3. Producers.

Rejectors simply reject any event likely to cause violation to occur; producers produce some other events whereas producers never reject, but calculate or derive something (new facts) for the end user. Producers and projectors can be subdivided.

BRG has divided business rules into:



Picture 1: Business Rule Types

Their classification has been described as follows: "A *structural assertion* is a statement that something of importance to the business either exists as a concept of interest or in relationship to another thing of interest. An *action assertion* is a statement that concerns some dynamic aspect of the business. It specifies constraints on the results that actions can produce. A *base fact* is a *fact* that is a given in the world and is remembered (stored) in the system. A *derived fact* is created by an inference or a mathematical calculation from *terms*, *facts*, other *derivations*, or even *action assertions* [0]". These types can be subdivided.

It is evident that there are lots of different definitions and classifications of the term 'business rule'. In this paper only some of them have been selected. One can say that business rules are a very important component of a business system because they have influence on activities to be performed and the system as a whole. More about business rules or classifications mentioned can be found in [0], [0] and [0]. The concept of fuzzy business rules, which relies upon fuzzy logic, has been presented in [0].

4. REACTIVE AGENTS

First we need to define and explain what an agent is, and then we are going to define a reactive one. The term "agent" has been (and still is) used very much and some very simple but also some extremely complex systems were treated and delineated as agent systems:

An agent is a computer system that is situated in some environment, and that is capable of autonomous action in this environment in order to meet its design objectives [0].

Agents are autonomous, persistent (software) components that perceive, reason, communicate and act in someone's favour, influencing its environment [0].

It's important to notice that different types of agents were introduced during the years; one can find terms like "desktop agent", "deliberative agent", "reactive agent", "vivid agent", "hybrid agent", "mobile agent" and so on; we have accepted the classification according to which two types of agents can be distinguished: deliberative and reactive; now we will define these two types.

One of the most popular architectures concerning agent systems is so called Belief/Desire/Intention (BDI) architecture. This architecture describes and determines the agent's behaviour according to information an agent possesses, goals which needs to be performed and reasoning about these goals and available information. Beliefs are sets of facts which represent the agent's knowledge about the environment. Goals are expressed as conditions over some interval of time and are described by applying various temporal operators to state descriptions; plans describe how an agent should react when certain facts are added to its belief database, or when it newly acquires certain goals [0]. This described architecture is an example for deliberative agents. So, deliberative approach subsumes reasoning based upon some knowledge base using some deduction technique. Some relationships between knowledge operator, belief operator, and desire operator in MASs are described in [0].

But researchers have come to the conclusion that reactivity is also a very important feature an intelligent agent should possess. Reactive paradigm became popular especially in autonomous robotics. It is hard to determine to what degree should an agent be reactive and to what deliberative, because this depends on lot of factors (tasks, architecture, application domain and so on) [0]. Reaction is used when there is no time for reasoning which is time consuming [0]. Reactive is suitable for dynamically changing environments performing an immediate response to some changes which have been recognized and perceived.

So, reactive agents react to the changes which have occurred and have been registered, and require some action (some events may occur, but actions are not needed). They don't possess the memory and internal state. In order to make experience and solve problems, they have to store information and remember. Reactive agents don't know anything about their environment so they don't possess any models of the environment which surrounds them. Reactive agents operate in presence and they will or will not react to certain stimuli [0]. We are not going to discuss the cooperation, coordination and communication, or issues like mobility or security; we refer to [0, 0, 0].

Although some people think (or thought) that "reactive agents" don't deserve much attention, it has been discovered and shown that many purely reactive agents can exhibit rather complex behaviour. A good example is a colony of ants; by following just a few simple rules the whole community exhibits very complex behaviour: they are able to find the food, to attack other ants and so on. The behaviour of a reactive agent is represented as a cycle with the following steps in [0]:

i) observe any input at time T,
ii) (optionally) record any such input,
iii) match conditions of condition-action rules with the inputs,
iv) (optionally) verify any remaining conditions of the rules using information in the knowledge base, using for steps (iii) and (iv) a total of R units of time,
v) select an atomic action which can be executed at time T+R+2 from the conclusions of rules all of whose conditions are satisfied,
vi) execute the selected action at time T+R+2, and (optionally) record the result, cycle at time T+R+3.

The connection between active databases and reactive agents is that both are able to perceive the environment and have to react to these changes, as has been already stated. Since the concept of reactivity is common to both fields, now we are going to see what can be done when combining these fields together.

5. IMPLEMENTATION

We can tell that reactive agents behave according to some set of rules and perform some actions on behalf of the creator. So, from this person's point of view, agents are operating according to this person's rules which are in fact business rules. So the connection between these three technologies is as follows: on the one side, active databases can be used for reactive agent's implementation and, on the other side, reactive agents operate according to some set of business rules. So, as can be concluded, if we want to implement a reactive agent within the active database, the indirect question in fact is whether different kinds of business rules can be implemented within the active database. Of course, this is not the only important question, but it is a crucial prerequisite for the successful implementation of a reactive agent within the active database. Having stated that it is easier to change the defined triggers instead of the written code in a programming language, the following step was to determine whether it is possible to implement a certain class of business rules which determine the agent's behaviour using an ADBMS instead of an application solution.

An attempt to compare agents and active databases was made; the result is presented in the Table 1. It is important to say that authors have put accent on BDI architecture and not on reactive agency and a lot of work has been done since than in active database theory field concerning different kinds of event, advanced transaction management, rule's actions and so on which offers new possibilities; that is why we want to discuss this idea any further. If somebody is more interested, we refer to [0].

Table 1: Active Database vs. Agent System [0]

	Active Database	Agent System	
	Events	Events	
Rules	Event Predicate	Invocation Condition	Plans
	Condition	Context Condition	
	Action	Plan Body	
	User Transactions	Intrinsic Goals	
	Database Facts	Beliefs	
	Update procedures	Goal-invoked plans	
	Integrity Constraints and Triggers	Event- or Fact-invoked plans	

As a model for defining business rules Date-s classification has been adopted. A class of business rules has been defined dictating under which conditions a student can take an exam at the Faculty of Organization and Informatics in Varaždin. As an idea a paper where

virtual university architecture is implemented using agents, as is presented in [0], has been very helpful. This class contains about 20 business rules, which, according to Date's business rules classification, fall into different types (presentation rules not being relevant in this context), and was implemented in form of an application which is actually used for exam registration. The question to be dealt with was whether it is possible to write (i.e., implement) all these business rules in an ADBMS instead of using an application version of these rules and what the advantages of such an approach would be. In determining whether such implementation was feasible, we intended to make the best use of the benefits offered by the ADBMS, and see which problems may arise in the process. Business rules could be grouped and these groups implemented by means of reactive agents. For instance, here is a list of some business rules:

R1: A student cannot take an exam if he has not enrolled on the course.

R2: An exam registration must be done at least 7 days before the exam is held.

R3: A student cannot register for the same exam on the same date twice.

R4: Registration of an exam a student has already passed is not valid.

R5: The exam registration date must be prior to the date when the exam was passed.

For their implementation PostgreSQL (an object-relational ADBMS) has been chosen. We have defined the set of triggers to be used as a means of implementing the class of business rules, and their behaviour was tested on real data. Since in this kind of application, called a 'notification application', action parts of triggers are mainly used to notice and raise exceptions, it was impossible for the rule execution not to be terminated. PostgreSQL has been tested and it has been detected that recursive rule execution can cause the system to restart. Thus one has to be very careful when recursion in triggering or activating graph occurs. Here is an example of a simple trigger which ensures that R5 is fulfilled (certain trigger definitions being huge):

```
CREATE TRIGGER Date_Checks BEFORE INSERT OR UPDATE ON exams
FOR EACH ROW EXECUTE PROCEDURE date_check();
```

This trigger uses the function *date_check()*, which has the following definition:

```
CREATE FUNCTION date_check() RETURNS opaque as '
        BEGIN
 IF new.reg_date>new.pass_date THEN
        RAISE EXCEPTION " The registration date is posterior to the date on
        which the exam is passed!";
END IF;
        RETURN NEW; END;'
LANGUAGE 'plpgsql';
```

The defined business rules class has been successfully implemented in PostgreSQL, although some problems concerning active functionality of the selected system have arisen.

When implementing reactive agents within the active database, many different types of events presented in active databases theory could be used in order to detect changes of interest; but we had some problems. Firstly, the ADBMS selected can detect only a small number of events, both simple and complex. If you need to detect some other complex events, it has to be programmed manually. Secondly, triggers management is definitely not a strong feature of PostgreSQL; triggers can be enabled or disabled, but they will be triggered whenever an event specified in the event part of the trigger occurs. Thirdly, different models of rules execution are not supported; the execution of some rule parts cannot be postponed. Nevertheless, one subset of the business rules defined was implemented without any problems, i.e. referential integrity. Finally, the actions which can be performed as a reaction to some event are also limited. These problems could perhaps have been avoided if another ADBMS had been selected.

Although the active database theory has developed significantly, practical solutions do not keep the pace with the theoretical results. There is an obvious gap between theory and practice. PostgreSQL is not the only system to have shown certain flaws in functionality, other systems have too. Thus if we had chosen another ADBMS, some other problems would probably have arisen due to the gap mentioned above. Nevertheless, we have come to the conclusion that it is easier to implement the defined business rules (reactive agents) using triggers since fewer variables are needed and the code to be written is shorter and easier to maintain, providing a more obvious solution. As a result, network traffic has also been reduced.

6. CONCLUSION

The aim of this paper was to determine and see whether active databases are a suitable means for implementation of reactive agents; indirectly this question was reformulated and as a prerequisite we had to see if it is possible to write different kinds of business rules within the active database. Namely, reactive agents act according to some set of business rules which have to be implemented when we try to build a reactive agent; all activities performed by an agent must obey certain business rules. The same concept on which reactive agents and active databases rely on and the idea of a virtual university have influenced on the set of business rules we have chosen. We have defined a class of business rules based on Date's classification of business rules. This class of business rules has been successfully implemented in an ADBMS PostgreSQL, although some problems requiring caution have been identified during the implementation phase and there are certain flaws in its functionality. Having checked the active features of some other ADBMSs, a conclusion can be drawn that although the active database theory has developed significantly, practical solutions do not keep the pace with the theory. So implementation problems have occurred, but were successfully resolved using some extended features of already mentioned ADBMS. The selected ADBMS is a restricted version of what could be called a 'full ADBMS'. Some active features, currently not supported in the selected ADBMS, would certainly make the implementation phase of some business rules, and thereby reactive agents, much easier. Therefore working with such a full ADBMS offering all the active features available at present would present yet another challenge. It is worth emphasizing that, from ADB point of view, a very big challenge is to design a full ADBMS. When discussing MAS in the context of reactive agents, business rules and active databases, issues like security, circumvented business rules and system design are still open and will be explored in future papers.

REFERENCES

[1] Andler, S. F., J. Hansson (1998): *Active, real time, and temporal database systems*, Springer, Berlin

[2] Bailey, J., A. Poulovassilis, P. T. Wood (2002): *Analysis and optimisation of event-condition-action rules on XML*, Computer Networks, vol. 39, no 3, pp. 239-259

[3] Bassiliades, N., I. Vlahavas (1997*): DEVICE: Compiling production rules into event-driven rules using complex event*s, Information and Software Technology, vol. 39, no 5, pp. 331-342

[4] Casati, F., M. Fugini, I. Mirbel (1999): *An environment for designing exceptions in workflows, Information Systems*, vol. 24, no 3, pp. 255-273

[5] Cetintemel, U., J. Zimmermann, Ö. Ulusoy, A. Buchmann (1999): *OBJECTIVE: a benchmark for object-oriented active database systems*, Journal of Systems and Software, vol. 45, no 1, pp. 31-43

[6] Chakravarthy, S. (1995): *Architectures and monitoring techniques for active databases: An evaluation*, Data & Knowledge Engineering, vol. 16, no 1, pp. 1-26

[7] Date, C. J. (2000): What not how – the business rules approach to application development, Addison Wesley, Reading

[8] Dinn, A., N. W. Paton, M. H. Williams (1999): *Active rule analysis and optimisation in the rock & roll deductive object-oriented database*, Information Systems, vol. 24, no 4, pp. 327-353

[9] Dittrich, K. R., H. Fritschi, S. Gatziu, A. Geppert, A. Vaduva (2003*): SAMOS in hindsight: experiences in building an active object-oriented DBMS*, Information Systems, vol. 28, no 5, pp. 369-392

[10] Eriksson E. Hans, Penker Magnus (2000): *Business modelling with UML: business patterns at work*, John Wiley & Sons, Canada

[11] Henry Hexmoor (2003): *Evolution of Agent Architectures*, W. Truszkowski, C. Rouff, M. Hinchey (Eds.): WRAC 2002, LNAI 2564, pp. 469-470, 2003., Springer-Verlag Berlin Heidelberg

[12] Ingo Stenge,Udo Bleimann and Jeanne Stynes (2003): *Social insects and mobile agents in a virtual university*, Campus-Wide Information Systems, vol. 20, pp. 84-89

[13] Jacek Malec (2001): *On Augmenting Reactivity with Deliberation in a Controlled Manner*, M. Hannebauer et al. (Eds.): *Reactivity and Deliberation in MAS*, LNAI 2103, pp. 76-91, Springer-Verlag Berlin Heidelberg

[14] Jacques Ferber (2001): *Multiagenten-Systeme, Eine Einführung in die Verteilte Künstliche Intelligenz*, Addison-Wesley

[15] James Bailey, Michael Georgeff, David Kemp, Davin Kinny, Kotagiri Ramamohanarao (1995): *Active databases and agent systems – a comparison*, Proceedings of the second international workshop on rules in database systems, Lecture notes in computer science 985, pp 342-356, Athens, Greece

[16] Kangsabanik, P., R. Mall, A. K. Majumdar (1997): *A Technique for Modelling Applications in Active Object Oriented Database Management Systems*, Information Sciences, vol. 102, no 1-4, pp. 67-103

[17] Koschel, A., P. C. Lockemann (1998): *Distributed events in active database systems: Letting the genie out of the bottle*, Data & Knowledge Engineering, vol. 25, no 1-2, pp. 11-28

[18] Lam, K.-Y, G. Law, V. Lee (2000): *Priority and deadline assignment to triggered transactions in distributed real-time active databases*, Journal of Systems and Software, vol. 51, no 1, pp. 49-60

[19] Maleković, M., M. Čubrilo (2003): *Knowledge, Belief, And Desire In Multi-Agent Systems*. CD Proceedings, The 7th IEEE International Conference on Intelligent Engineering Systems, INES 2003, pp. 100-103.

[20] Martin Riedmiller, Andrew Moore, Jeff Schneider (2001): *Reinforcement Learning for Cooperating and Communicating Reactive Agents in Electrical Power Grids*, M. Hannebauer et al. (Eds.): *Reactivity and Deliberation in MAS*, LNAI 2103, pp. 137-149, 2001., Springer-Verlag Berlin Heidelberg

[21] Montesi, D., E. Bertino, M. Bagnato (2003): *Refined rules termination analysis through transactions, Information Systems*, vol. 28, no 5, pp. 435-456

[22] Montesi, D., R. Torlone (2002): *Analysis and optimization of active databases, Data & Knowledge Engineering, vol. 40, no 3, pp. 241-271*

[23] Pai, A. V., R. F. Gamble, R. T. Plant (1999): *Using KBS verification techniques to demonstrate the existence of rule anomalies in ADBs*, Information and Software Technology, vol. 41, no 10, pp. 627-638

[24] Paton, N. W. (1998): *Active rules in database systems*, Springer, New York

[25] Robert Kowalski, Fariba Sadri (1997): *An Agent Architecture that Unifies Rationality with Reactivity*, Department of Computing, Imperial College

[26] Rosane Maria Martins, Magali Ribeiro Chaves, Luci Pirmez and Luiz Fernando Rust da Costa Carmo (2001): *Mobile agents applications*, Electronic Networking Applications and Policy, vol. 11, no 1, pp. 49-54

[27] Ross, R. G. (2003): *Principles of the business rule approach*, Addison Wesley, Boston

[28] Saygin, Y., Ö. Ulusoy, S. Chakravarthy (1998): *Concurrent rule execution in active databases*, Information Systems, vol. 23, no 1, pp. 39-64

[29] Tan, C.-W., A. Goh (1999): *Composite event support in an active database*, Computers & Industrial Engineering, vol. 37, no 4, pp. 731-744

[30] Thalhammer, T., M. Schrefl, M. Mohania (2001): *Active data warehouses: complementing OLAP with analysis rules*, Data & Knowledge Engineering, vol. 39, no 3, pp. 241-269

[31] V. Marik et all. (2002): Multi Agent Systems & Applications, Springer-Verlag Berlin Heidelberg

[32] ***: *Building a Business Rules System*, <http://www.dmreview.com>

[33] ***: *Defining Business Rules ~ What Are They Really?*, <http://www.businessrulesgroup.org/first_paper/br01c0.htm>

[34] ***: *Doorsey, P.: What are business rules?*, <http://www.dulcian.com/BRIM%20Documents/What%20Are%20Business%20Rules.htm.>

[35] ***: *PL/pgSQL - SQL Procedural Language*, <http://www.postgresql.org/docs/7.2/static/plpgsql.html>