A Structure of Semantic Service in a Distributed Knowledge Based System

Nataliia Kulykovska[1][0000-0003-4691-5102], Artur Timenko[2][0000-0002-7871-4543]

[1]Zaporizhzhia National Technical University, Zhukovsky str., 64,Zaporizhzhia, 69063, Ukraine
natalya.gontar@gmail.com
[2]Zaporizhzhia National Technical University, Zhukovsky str., 64,Zaporizhzhia, 69063, Ukraine
timenko.artur@gmail.com

Abstract. The main difference in distributed systems based on knowledge is the use of the service approach and ontologies in knowledge engineering. Semantic service acts as a software agent that regulates the interaction of all components of the system. Semantic service provides: semantic principle, which defines a formal description of information and allows you to define the following characteristics of services: scalability, semantic interoperability, formal models of services and ontologies; the principle of decision making; the principle of distribution, which allows you to aggregate the capabilities of several computing objects through cooperation. The communication and collaboration mechanism of the distributed systems based on knowledge covers the message transfer between services, information exchange between semantic service and distributed knowledge bases, and the semantic service behaviors in the entire process from request submission, request handle, to result return. The knowledge base is formed from three types of ontologies. The rules for recognizing performance problems are converted into ontology and the system knowledge base is placed. The working memory of a semantic service contains facts that correspond to simple events, composite events, and identified problems and recommendations. The algorithm for using semantic service is discussed in the article. To assess the performance of the SS attached, create an information system of two components: event trace generator and event trace analyzer.

Keywords: distributed system, semantic service, components of a distributed system, knowledge engineering, data, ontology, services, semantic web, structure of semantic service

1 Introduction

Economic globalization and global networks have become the trend of world development. Especially with the development of information technology and the innovation of management theory, modern enterprises need to go beyond the boundaries of traditional companies to achieve for a rapid and effective integration of resources on a global scale, thus can produce high-quality products that meet the needs of the market or provide services the customer need in a very short period of time [1].

In this case, a new distributed computer system in terms of development of modern technologies put the emphasis on the properties of interoperability and scalability. This direction is connected with the rapid growth of blockchain technology and the Internet of things (IoT). As analysts predict that, by 2025, the share of block chain apps will account for 10% of world gross domestic product [2]. By 2020 will create more than 30 billion IoT devices. IoT will affect every industry, from retail to health care [3].

At the moment in the literature there are a large number of definitions of the concept of "distributed system". The most comprehensive definition proposed by AS Tanenbaum [4]: “Distributed system (DS) is a set of independent computers, which is perceived by its users as the only consistent system.” Another definition is proposed in [5]: DS are software and hardware systems, in which execution of operations (actions, calculations) necessary to ensure the target functionality of the system is distributed (physically or logically) between different performers. In the computing field, under our computer, in our study, we will understand the software and hardware system created for a specific practical application, the functionality of which is distributed on various nodes.

On the one hand, DS are tools that allow solving a large number of complex tasks, most of which are not solved by other methods. DS can eliminate the main drawback of centralized systems - the limitations of increasing computing power. At the same time, the analysis revealed a number of problems that need to be solved. The limited use of DS is complicated by the use of equipment from different manufacturers with different types of architectures. Due to the wide variety of aspects of building computing systems, as well as the variety of existing operating systems, it becomes necessary to create methods for adaptive planning of distribution of flows in a distributed system, which will significantly speed up the processing of incoming requests for services and increase the overall system performance.

Modern representations of data are changing the ways and forms of communication, production and consumption of information [6]. The dominance of horizontal relations, structure-role of information decentralization of all types of data available at any time on every device. The user should not care about the specific technology used to provide computing capacity or data storage, so you can say that a user has some information about the remote resource. Distributed systems based on knowledge (DKBS) in order to study the data, their processing and use evolving technologies of the semantic web.

The formal model of the DKBS consists of a set of ontologies; lots of services; a set of events that describe the processes of the system; semantic service (SS); a set of composite services and knowledge base. The completeness and effectiveness of the system is determined by a multitude of ontologies. The main difference in DKBS is the use of the service approach and ontologies in knowledge engineering. SS acts as a software agent that regulates the interaction of all components of the system.

2 Literature review

Software agents originally were discussed in the 70's, and in the mid 90's briefly gained some momentum but then stalled. The "software agent" term has found its way into a number of technologies and has been widely used, for example, in artificial intelligence, databases, operating systems and computer networks literature. Although there is no single definition of an agent [7, 8, 9] all definitions agree that an agent is essentially a special software component that has autonomy that provides an interoperable interface to an arbitrary system and/or behaves like a human agent, working for some clients in pursuit of its own agenda. Most discussions on agents focus on their autonomy, intelligence, mobility and interaction [10, 11, 12, 13, 14]. Agent-based systems [15, 16] claim to be next generation software capable of adapting dynamically to changing business environment and of solving a wide range of knowledge processing application. Although sophisticated software agents can be difficult to build from scratch due to the skills and knowledge needed, the widely available agent construction toolkits may provide a quick and easy start to building software agents without much agent expertise. Significant research and development into multi-agent systems (MAS) has been conducted in recent years [17, 18, 19], and there are many architectures available today [20, 21]. Nevertheless, several issues still need to be faced to make the multi-agent technology widely accepted: secure and efficient execution supports; standardization; appropriate programming languages and coordination models.

Decentralization and openness are inherent properties of multiagent systems (MAS). The technologies they provide are thus the right abstraction for developing Web-oriented applications. Moreover, different works have been proposed to use Semantic Web technologies (SWT) for representing various dimensions of MAS (e.g., interaction protocols, norms, organizations).

Consequently, recent research in MAS have seen an intensive use of Knowledge representation together with increasing use of SWT. We envision that SWT will ultimately play a central role in all parts of MAS. Thus far, work combining MAS and SWT have only been concerned about addressing one dimension of MAS at a time. Also, they were mostly tackling the agent [22, 23] and interaction dimensions [24, 25] to ease communications, especially on domain knowledge. Other works have used those technologies to model part of the organization structure [26], norms and commitment [27], reputation [28] and more. The situation shows that it is time to go beyond these ad hoc solutions and integrate the pieces into a complete Semantic-Web-based infrastructure. We observe too that none of the mentioned contributions were really taking advantage of the Web aspect of these technologies, except some Web service integration. We also want to provide models and specifications for the SS so that services can uniformly query and reason about DKBS, web services, data, and ontologies.

3 A Distributed Knowledge Based System

The proposed architecture is based on services and is designed to support all the processes of the system life cycle. That is, the proposed architecture is meant to include all the concepts necessary to perform all activities related to the life cycle. We shall adopt a generic, knowledge-based architecture. The enterprise members are represented by autonomous agents, geographically scattered, which are able to cooperate to achieve a common business goal.

The various persons constituting the DKBS, in order to process distributed knowledge bases, assume the following roles:

- Knowledge Manager: is in the top of the hierarchy, and acts as a project manager, but in higher levels. It's like a knowledge strategist, cooperating, defining, and distributing the knowledge to coordinate all the other roles;
- Knowledge Provider: is the owner of human knowledge. It is typically an expert in the application domain, but could be another person in the organization who does not have the expert status;
- Knowledge Analyst: uses a range of methods and tools that make the analysis of a standard knowledge-intensive task relatively straightforward;
- Knowledge System Developer: is responsible for the design and implementation. The developer must have a basic background of analysis methods. In knowledge system development, the main knowledge problems have been solved by the knowledge analyst. Therefore, this role must have some skills of software designers;
- Knowledge User: makes use, directly or indirectly, of a knowledge system. Its interaction with the knowledge base system is important for the project development and validation;
- Project Manager: manages the project, specially the knowledge engineer and the knowledge system developer.

In turn, the structure of the system based on knowledge, is shown in Fig. 1. The main components are [29, 30,31]:

- base of knowledge. Base of knowledge intended for storage of knowledge about the subject. Its concrete form depends strongly on the chosen model of knowledge representation. The presence of this component is the main difference between the systems based on knowledge from other types of programs;
- output machine. Output machine generates a response to the user request using the knowledge base. Its principle of operation is also dependent on the chosen model of knowledge representation;
- editor knowledge base – program to change the contents of the knowledge base;
- user interface – the mechanism by which the communication user and the system.

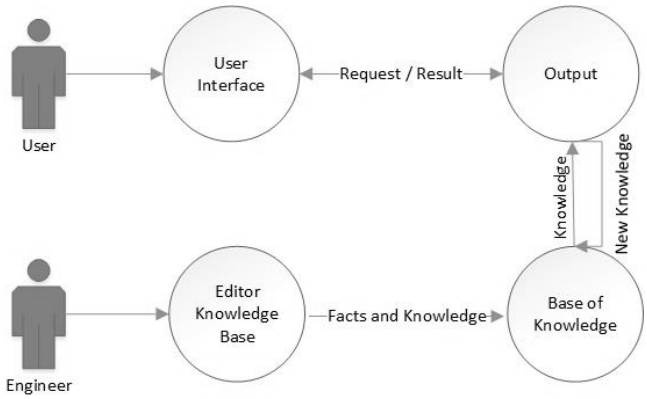

Fig. 1. Structure of systems based on knowledge.

In addition to the agents representing the distributed knowledge base, a SS is introduced in the MAS community. This agent reacts by seizing deal opportunities present in the DKBS, and proceed thus to establish the corresponding virtual entities.

The principle of the Distributed Knowledge Base proposed in this paper is to depends on a number of ontologies that provide semantic principle: ontologies of events DKBS (Task Ontology); ontologies of services (Application Ontology); ontology specialized areas of data systems based on (Domain Ontology). (fig. 2).

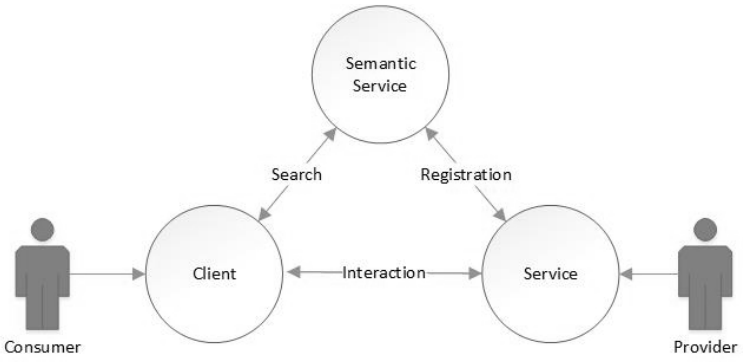

Fig. 2. Structure of DKBS.

DKBS can be represented as a model of interaction of the main artifacts: clients, services and SS. Each component in the system is associated with a specific ontology. SS provides:

- semantic principle, which defines a formal description of information and allows you to define the following characteristics of services: scalability, semantic interoperability, formal models of services and ontologies;
- the principle of decision making;

- the principle of distribution, which allows you to aggregate the capabilities of several computing objects through cooperation.

Considering all the types of ontologies [32, 33, 34], we chose it for three, because the Application Ontology is a model of a service describes its functional and non-functional characteristics. Domain Ontology is a certain base set of knowledge for each subject area. Task Ontology keeps information about all the functions and actions of the system.

Each ontology has its own expressive capabilities depending on their functional purpose. Ontology-oriented for subject area (Domain Ontology), describe the dictionary of terms concrete and formal set four end subsets: concepts, relations, axioms and interpretation functions. Task Ontology consists of a Glossary of terms, specialize tasks and actions in subject area. Every task has a different status and stages of its implementation. The main feature for all concepts of the ontology problem is the time. Application Ontology is the most specific ontology, in addition to all the basic concepts, contains specialized terms and instances subject area.

SS includes domain ontology and a set of modules to operate the services and their ontological description a lot of events we have formed as a set of claims that can be applied to DKBS and a set of axioms of their appearance.

4 Structure of Semantic Service in a DKBS

In each interaction of the SS, the ontology functionality is necessarily preserved. Each function is characterized by the input (1) and output arguments (2). Correspondingly, each action SS (3) is characterized by a function of the work with the ontology, the values of its arguments when you call and the completion of the system.

$$f_{SS} \to FA_k^{IN} = (fa_1^{IN},..., fa_k^{IN}); k = 1,..., K. \quad (1)$$

$$f_{SS} \to FA_n^{OUT} = (fa_1^{OUT},..., fa_K^{OUT}); k = 1,..., K. \quad (2)$$

$$a_{ij} = \{f_k, FAval_i^{IN}, FAval_j^{OUT}); i = 1,..., K; j = 1,...K; f_k \in F. \quad (3)$$

In the DKBS, the communication and collaboration mechanism between services is a problem that must be well considered in the design of the SS structure. The communication and collaboration mechanism covers the message transfer between services, information exchange between SS and distributed knowledge bases, and the SS behaviors in the entire process from request submission, request handle, to result return. These procedures can be summarized as follows:

- SS deploys appropriate services on its local Knowledge Base, performs the registration on the knowledge bases;
- The retrieval request accesses the system periodically. SS gets retrieval requests and searches its Knowledge Base.
- SS returns the retrieval result to the system after the search finishes.

- The SS implies a classification and a sort order to the retrieval result according to its corresponding retrieval request.
- User views the retrieval results on the interface and selects the needed Knowledge. With the information about the Knowledge Base embedded in the retrieval result, the user then contacts the relevant enterprise to obtain the Knowledge.

The principle of operation of the SS is illustrated in Fig. 3. The knowledge base is formed from three types of ontologies. The rules for recognizing performance problems are converted into ontology and the system knowledge base is placed. The working memory of a SS contains facts that correspond to simple events, composite events, and identified problems and recommendations. The algorithm for using SS to perform an analysis is as follows. The initial data is the sequence of events of the system, let's call it the trace. Trace contains simple events. *E* The algorithm processes the events of the route in the order of the time of their occurrence and performs the following cycle:

1. Read the next event from the trace *E*.
2. Convert it to the concept of ontology and add it to the working memory of the SS.
3. Start the SS. If the new event is a search query, then pick up the relevant answers in the knowledge base. For an indefinite concept of a simple event, the output engine consistently executes the rules for constructing composite events, for constructed fact-composite events, the rules for identifying performance problems. The result of triggering a SS can be:
   (a) adding a new fact-simple event to some of the constructed fact-composite events;
   (b) identification of fact-composite events of working memory for compliance or non-compliance with the performance problem;
   (c) delete or save a new event in the working memory;
   (d) search for a saved event.
4. Retrieve the search results from the working memory or offer recommendations to solve the problem.

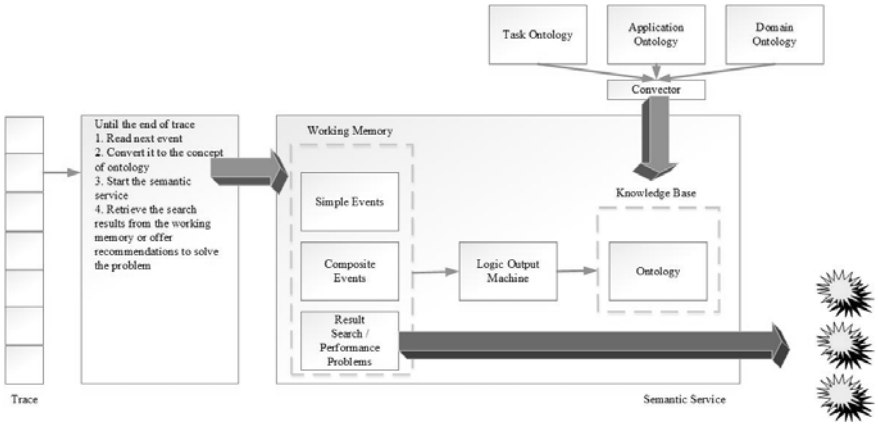

Fig. 3. Structure of Semantic Service.

To assess the performance of the SS attached, create an information system of two components:

- event trace generator DKBS;
- event trace analyzer for SS.

Events are recorded in the track when calling action functions. A simple event is represented as:

$$e = \langle f, et, EPval, t, d, cs \rangle, \quad (4)$$

where $f \in F$ - action function;

et - the type of event that sets its parameters $et \Rightarrow EP = (ep_1, ..., ep_K)$;

$EPval = (v_1, ..., v_K)$ - event parameter values;

t - time of the event;

d - event duration;

cs - service code.

The implementation scheme of the tracer generator is presented in fig. 4. The tracer generator consists of the following components:

- *OntologyGenerator* program converts messages into ontology concepts;
- *library SM* is designed to track messaging between services;
- *library Timer* provides functions for obtaining current system time with high accuracy;
- *library TracerWriter* saves simple events created by the tracer to a trace file.

Presumably all generator modules are implemented in the Java programming language.

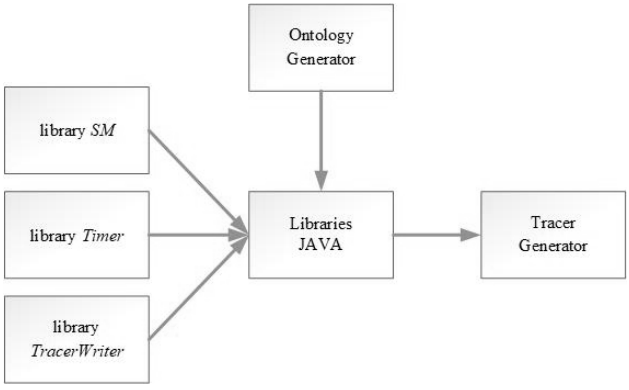

Fig. 4. The tracer generator.

The implementation scheme of the SS trace analyzer is shown in fig. 5. The trace analyzer consists of the following components:

- *library UI* implements the CC user interface.
- *library TraceReader* provides reading simple events from trace files.
- *library Ontology* implements event conversion to ontology concepts.
- *library Query* generates queries to the SS.

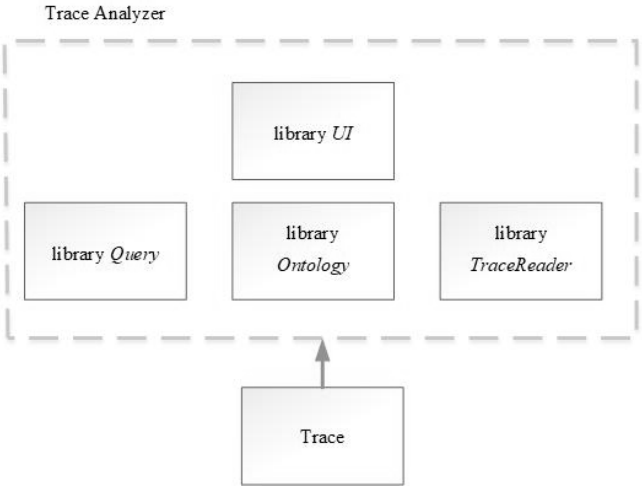

Fig. 5. The trace analyzer.

5 Conclusion

Modern representations of data are changing the ways and forms of communication, production and consumption of information. The dominance of horizontal relations, structure-role of information decentralization of all types of data available at any time on every device. The user should not care about the specific technology used to provide computing capacity or data storage, so you can say that a user has some information about the remote resource. DKBS in order to study the data, their processing and use evolving technologies of the semantic web. The formal model of the DKBS consists of a set of ontologies; lots of services; a set of events that describe the processes of the system; SS, a set of composite services and knowledge base. The completeness and effectiveness of the system is determined by a multitude of ontologies. The main difference in DKBS is the use of the service approach and ontologies in knowledge engineering. SS acts as a software agent that regulates the interaction of all components of the system. SS provides: semantic principle, which defines a formal description of information and allows you to define the following characteristics of services: scalability, semantic interoperability, formal models of services and ontologies; the principle of decision making; the principle of distribution, which allows you to aggre-

gate the capabilities of several computing objects through cooperation. The communication and collaboration mechanism of the DKBS covers the message transfer between services, information exchange between SS and distributed knowledge bases, and the SS behaviors in the entire process from request submission, request handle, to result return. The knowledge base is formed from three types of ontologies. The rules for recognizing performance problems are converted into ontology and the system knowledge base is placed. The working memory of a SS contains facts that correspond to simple events, composite events, and identified problems and recommendations. The algorithm for using SS is discussed in the article. To assess the performance of the SS attached, create an information system of two components: event trace generator and event trace analyzer.

References

1. Ruzhi Xu, Peiguang Lin, Cheng Liu: Research on Distributed Knowledge Base System Architecture for Knowledge Sharing of Virtual Organization. Atlantis Press, pp. 349-357 (2010)
2. WEB 3.0 budet pitat'sya stekom tekhnologii blokcheyn [WEB 3.0 will be powered by blockchain technology stack]. Available at: https://101blockchains.com/ru/web-3-0/ (accessed 01.03.2019)
3. Deloitte projects by 2025 that 10% of global GDP to be built on blockchain applications. Available at: https://www.freightwaves.com/news/2017/9/2/deloitte (accessed 02.09.2017)
4. Tanenbaum A., Van Steen M.: Distributed systems. Pearson Prentice Hall (2007)
5. Burns B.: Designing Distributed Systems: Patterns and Paradigms for Scalable, Reliable Services. O'Reilly Media, Inc (2018)
6. Allemang D, Hendler J.: Semantic web for the working ontologist modeling in RDF, RDFS and OWL. Elsevier Inc. (2008)
7. Russell, S., Norvig, P.: Artificial Intelligence: A Modern Approach. Prentice-Hall, Englewood Cliffs (1995)
8. Genesereth, M.R., Ketchpel, S.P.: Software Agents. Communications of the ACM 37(7): 48–53 (1994)
9. Wooldridge, M.J., Jennings, N.R.: Intelligent Agents: Theory and Practice. Knowledge Engineering Review 10(2): 115–152 (1995)
10. Stoian, G., Popirlan, C.I.: A proposal for an enhanced mobile agent architecture (EMA). In: Annals of the University of Craiova, Mathematics and Computer Science Series, vol. 37(1), pp. 71–79 (2010)
11. Tandareanu, N., Popirlan, C.I.: A Mobile Agents Approach for Knowledge Bases Processing. In: Samarati, P., Yung, M., Martinelli, F., Ardagna, C.A. (eds.) ISC 2009. LNCS, vol. 5735, pp. 27–32. Springer, Heidelberg (2009)
12. Popirlan, C.I.: Knowledge Processing in Contact Centers using a Multi-Agent Architecture. WSEAS Transactions On Computers 9(11), 1318–1327 (2010)
13. Popirlan, C.I., Stefanescu, L.: Mobile Agents System for Intelligent Data Analysis. In: Proceedings of WSEAS Applied Computing Conference (ACC 2009), Athens, Greece, pp. 663–668 (2009)
14. Popirlan, C.I., Dupac, M.: An Optimal Path Algorithm for Autonomous Searching Robots. In: Annals of University of Craiova, Mathematics and Computer Science Series, vol. 36(1), pp. 37–48 (2009)

15. Sycara, K.P.: Multiagent Systems. AI Magazine, 79–92 (1998)
16. Wooldridge, M.: Introduction to Multiagent Systems. John Wiley and Sons, UK (2002)
17. Kabbaj, A.: Development of Intelligent Systems and Multi-Agents Systems with Amine Platform. In: Schärfe, H., Hitzler, P., Øhrstrøm, P. (eds.) ICCS 2006. LNCS (LNAI), vol. 4068, pp. 286–299. Springer, Heidelberg (2006)
18. Balachandran, B., Enkhsaikhan, M.: Developing Multi-agent E-Commerce Applications with JADE. In: Apolloni, B., Howlett, R.J., Jain, L. (eds.) KES 2007, Part III. LNCS (LNAI), vol. 4694, pp. 941–949. Springer, Heidelberg (2007)
19. Rodrguez, S., Fernndez, V., Julin, V., Corchado, J.M., Ossowski, S., Botti, V.: A THOMAS Based Multi-Agent System for Recommendations and Guidance in Malls. Journal of Physical Agents 3(2), 21–26 (2009)
20. Hagras, H., Callaghan, V., Colley, M.: Intelligent Embedded Agents. Journal of Information Sciences, 289–292 (2005)
21. Bouchachia, A., Prossegger, M.: A Bi-Clustering Agent-based Approach for Map Segmentation. In: Proceedings of the IEEE Symposium on Intelligent Agents, IA 2009, Part of the IEEE Symposium Series on Computational Intelligence, pp. 99–105 (2009)
22. T. Klapiscak and R. Bordini. JASDL: A Practical Programming Approach Combining Agent and Semantic Web Technologies. In Proc. of DALT, pp 91–110, (2009)
23. M. Hadzic, E. Chang, and P. Wongthongtham: Ontology-Based Multi-Agent Systems. Springer (2009)
24. OWL-DL as a FIPA-ACL content language. In Proc. of FOCA 2006, (2006).
25. Y. Zou, T. Finin, Y. Peng, A. Joshi, and R. S. Cost: Agent Communication in DAML World. In Proc. of WRAC 2002, pp. 347–354 (2002)
26. D. Okouya, L. Penserini, S. Saudrais, A. Staikopoulos, V. Dignum, and S. Clarke: Designing MAS Organisation through an Integrated MDA/Ontology Approach. In Proc. of TWOMDE: pp. 55–60 (2008)
27. N. Fornara and M. Colombetti: Representation and monitoring of commitments and norms using OWL. AI Comm., 23(4):341–356 (2010)
28. L. Nardin, A. Brand˜ao, and J. Sichman: Experiments on semantic interoperability of agent reputation models using the SOARI architecture. Eng. Appl. of AI, 24(8), pp. 1461–1471 (2011)
29. Dzharratano Dzh.: Ekspertnyye sistemy: printsipy razrabotki i programmirovaniye: per. s angl. K. A. Ptitsyna. – 4-ye izd. [Expert systems: principles of development and programming]. Moskva, Vil'yams (2007)
30. Smolin D. V.: Vvedeniye v iskusstvennyy intellekt: konspekt lektsiy [Introduction to Artificial Intelligence]. Moskva (2004)
31. Behnam Azvine, Nader Azarmi, Detlef D. Nauck: Intelligent Systems and Soft Computing: Prospects, Tools and Applications. Springer (2006)
32. Gruber Thomas R.: A translation approach to portable ontology specifications. Appeared in Knowledge Acquisition, vol. 5(2), pp. 199-220 (1993)
33. Yu Liyang: Introduction to Semantic Web and Semantic Web services, CRC Press (2007)
34. John Davies, Rudi Studer, Paul Warren: Semantic Web technologies: trends and research in ontology-based systems. John Wiley & Sons (2006)