Adaptation of the Jena Framework for Fuzzy Reasoning in the Semantic Web Expert System

Olegs Verhodubs

November 2, 2023

Adaptation of the Jena Framework for Fuzzy Reasoning in the Semantic Web Expert System

Olegs Verhodubs

oleg.verhodub@inbox.lv

Abstract— **The main purpose of this paper is to present a way to adapt the Jena framework for fuzzy inference. This is necessary, because the Jena framework has no built-in way to infer using fuzzy values. The Jena framework with the ability of fuzzy inference implies as part of a Semantic Web Expert System, which is being designed to use OWL (Web Ontology Language) ontologies from the Web, to generate rules from these ontologies and to supplement or even to develop its knowledge base in automatic mode. Available publications show that the problem of the Jena framework adaptation for fuzzy inference is not investigated deeply enough.**

Keywords*— *Semantic Reasoner, Fuzzy Reasoning, Fuzzy Jena, Semantic Web, Expert Systems

I. INTRODUCTION

Nowadays the Semantic Web technologies give new impetus to the development of the Web. This impetus was mainly based on the use of semantics and was prepared by technological development of the Web during the last few decades, when the users of the Web were provided with access to far more information than could be comprehended or managed effectively [1].

Semantics as a base of the Semantic Web is expressed in ontologies. Ontologies are defined as explicit and formal specifications of a conceptualization of a domain of interest [2] and consist of concepts (or classes), relations, instances and axioms. The use of ontologies has a lot of advantages. For example, ontologies enable to organize and to find information based on meaning, not just text. In addition, the use of ontologies improves the presentation of information, and it means that results can be clustered by meaning, but not in linear way. The use of ontologies can also make the task of information integration doable. Early work in different countries on defining ontology languages have led to the development of OWL (Web Ontology Language) by the W3C (World Wide Web Consortium). OWL builds on the RDF (Resource Description Framework), which is a data modeling language based on triples: subject, predicate and object [1]. In fact, OWL has three species: OWL Lite, OWL DL and OWL Full. OWL Full is a superset of OWL DL, and OWL DL is a superset of OWL Lite.

The number of OWL ontologies located in the Web is large, what can be verified using Watson Semantic Web Search (watson.kmi.open.ac.uk). Moreover, interest in the representation of information in the form of OWL ontologies is not waning. The development of the second version of OWL namely OWL 2 [3] and a lot of conferences, which were dedicated to the Semantic Web and the Semantic Web technologies, where OWL was a cornerstone of numerous researches, came upon the idea of using OWL ontologies in the subarea of artificial intelligence known as expert systems. In this context, ontologies are seen as a widening resource that cannot be unused. The research of OWL ontology transformation to rules [4] became a key research and allowed to formulate the final goal that is to develop a Semantic Web Expert System (SWES), which will be capable to use OWL ontologies from the Web, to extract rules from these ontologies and to supplement its knowledge base with the extracted rules in automatic mode [5]. To achieve the final goal of research and to develop SWES it is necessary to solve several tasks. The main purpose of this paper is the presentation of the way to adapt the Jena framework for fuzzy inference. This is necessary, because the Jena framework has no built-in way to infer using fuzzy values, but this ability is a backbone since the SWES is aimed to work with fuzzy rules. Analysis of the available sources reveals that the problem is not solved either for SWES, or to any other system.

This paper is structured as follows. Section 2 describes the reasons for the need of fuzzy Jena namely why it is necessary to adapt the Jena framework for fuzzy inference and also reflects the requirements for fuzzy Jena. Section 3 represents the proposed realization of fuzzy Jena. And finally some conclusions are outlined.

II. REQUIREMENTS FOR FUZZY JENA

The Web is an environment, filled with a lot of information, produced by different people for different purposes. In this connection it is logical that diversity of information producers and their purposes do not allow treating the information in the same way in the sense of belonging to a particular subject area. Errors and human factor only exacerbate the problem. Hence SWES as an expert system, which is keen to work with the Web sources, has to dispose the mechanism to cope with uncertainty in the Web. Despite the fact that expert systems are quite new field of artificial intelligence, there are several ways to manage this uncertainty. Here they are [6]:

- Conditional probability,
- Trust coefficients,
- Fuzzy sets and fuzzy logic,
- Possibility theory.

Fuzzy sets and fuzzy logic have such advantages over other methods as scientific validity, efficiency in implementation, and also the fact that they have many previous examples of successful use. That is why fuzzy sets and fuzzy logic are chosen for realization in the SWES. Fuzzy sets are helpful in the formation of resource base for rule generation that is in the process of ontology merging [7]. Thus as a result of merging process the single ontology is obtained, where all necessary for rule generation ontology elements have fuzzy values. Fuzzy values of ontology elements give an opportunity to generate fuzzy rules [7], and this is a serious problem. The fact is that the Jena framework, which is chosen for use in SWES, does not allow inferring using fuzzy rules. It is possible that one of the main reasons of this is the lack of an official standard for displaying fuzzy ontology. Despite the lack of fuzzy inference, the Jena framework is chosen due to a number of advantages as type of licensing, good reputation of developer, availability of detailed documentation, focus on Java programming language, presence of ontology consistency checking function, having multifunction capabilities of reasoning on OWL ontologies, availability of rule support and reasoning on these rules, and also having storage subsystem [8]. Plenty of advantages and only one severe disadvantage drive at the idea of adaptation of the Jena framework to fuzzy inference. To implement this idea, it is necessary to work out requirements for the Jena framework with the function of fuzzy inference (Fuzzy Jena).

In general, the flow of information within the SWES can be represented as follows (Fig. 1.):

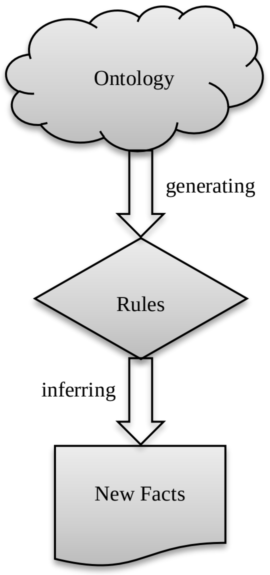

Fig. 1. The flow of information in SWES.

Each element of the SWES information flow has to be adapted for fuzzy inference. As can be seen, the first element is ontology (Fig. 1), and it refers to the OWL ontology here. The specification of OWL has several advantages, but its main disadvantage for fuzzy inference is in the impossibility of storage for fuzzy values by means of specially designed for this purpose constructs. Certainly, there are some attempts to extend the OWL specification in the direction of fuzziness (for example, [9]), but these attempts are not official and hence the extentions cannot be used without problems because of lack of continuity. One more problem, which is associated with OWL use for storage of fuzzy values, refers to the implementation of the SWES. The task of OWL ontology transformation to rules or it is better to say to the concept map [11], where rules are coded in abstract IF…THEN form, is realized in such a way that each property, class or relation influences the condition or result part of any rule. Therefore it is not possible to set fuzzy values by means of these ontology elements without loss of meaning. It is necessary such a way to set fuzzy values, at which fuzzy values will be kept separate from the main elements of OWL ontology. This problem is partly solved in the research of ontology merging for the SWES, where values of membership functions, obtained in the process of ontology merging, are stored in OWL ontology comments [7]. One of the complexity of this manner is an objective limitation of OWL ontology means of expression, which do not allow to assign comments to relations or in other words to object properties in terms of OWL specification [12]. Of course, coding fuzzy values in OWL ontology, it is necessary to remember that these fuzzy values have to be able to be read by the Jena framework. The second element of the SWES information flow is a set of rules (Fig.1.). The Jena framework supports the inference based on rules in a specific format of the Jena framework [8]. The main requirement for rules in the Jena framework is the ability to infer the result with a degree, expressed by a real number. Of course, it is also necessary to have an ability to store and to extract the result of inference, but this requirement refers mostly to the last element of the SWES information flow that is to the new generated facts (Fig. 1.). One more important task is to combine the capabilities of the OWL specification, the capabilities of the Jena framework regarding rules and inference and the capabilities of organizing new facts, which are generated by the Jena framework.

III. Realization of Fuzzy Jena

The structure of any system is decisive for its functioning that is why it is necessary to overview the structure of the SWES before adapting the Jena framework to fuzzy inference. SWES is designed as an expert system, and it means that its structure is similar to the structure of any other expert system [5]. Therefore there is no need to repeat and to consider the whole structure of the Semantic Web Expert System here. However the SWES knowledge base has to be examined in more detail, because it is the most closely associated with an inference engine SWES part and, therefore, it strongly influences the overall process of inferring.

Let us remind that the knowledge base of the Semantic Web Expert System receives OWL ontologies from the Web by means of searching them according to a user's request. Further these OWL ontologies are stored in the OWL repository. After that the stored OWL ontologies from the OWL repository are merged into a single OWL ontology, and it is placed in the appropriate repository for the merged ontology. Then this merged ontology is transformed to rules in abstract form. These rules are stored in the appropriate repository named as the concept map. After that the rules from the concept map are transformed to the rules in the format of the Jena framework; the Jena rules are stored in the appropriate repository, too. Finally the rules in the format of the Jena framework are supplied to the inference engine [7].

Thus, it is possible to conclude that the knowledge base of the SWES is divided into several storages. These storages are the following:

- Storage for OWL ontologies (OWL Repository),
- Storage for merged ontology,
- Storage for concept map,
- Storage for Jena rules.

So, the structure of the SWES knowledge base, described in the previous paper [7] looks like as follows (Fig.2.):

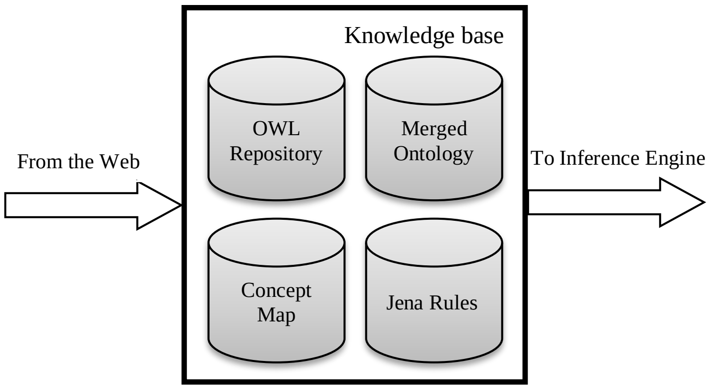

Fig. 2. SWES knowledge base.

At first glance, the SWES knowledge base, which is shown in Fig. 2, seems to be exhaustive, but it is not the case. Indeed, it is expected that in the process of inferring new facts will be produced, and this will be the main profit of the Semantic Web Expert System. However it can be noticed that there is no explicit storage for placing produced new facts in the SWES knowledge base (Fig.2.). Furthermore, there is no explicit storage for placing the facts, which are obtained from a user, whereupon the SWES starts. On this basis, it is possible to suppose that the structure of the SWES knowledge base requires some changes. These changes should be directed towards the separation of the knowledge base into two areas of TBox and ABox in terms of description logic terminology, where the TBox contains the axioms defining the classes and relations in an ontology, while the ABox contains the assertions about the individuals in the domain [12]. So, considering this fact, the SWES knowledge base can be represented as follows (arrows “From the Web” and “To inference engine” are not displayed because of place constraints):

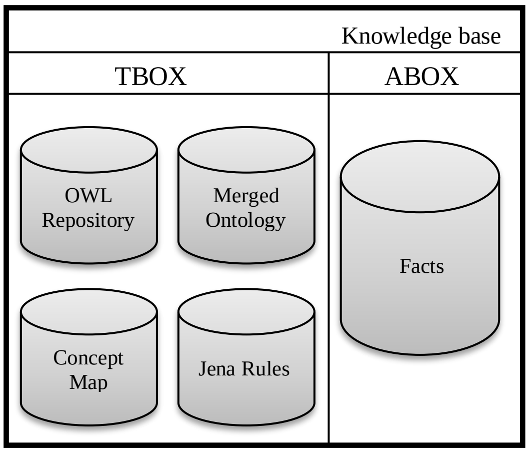

Fig. 3. Detailed structure of SWES knowledge base.

In Fig.3, it can be seen that there is only one storage for placing facts. This applies to the facts, which are entered by a user and the facts, which are produced by the SWES inference engine. This provides a logical data integrity of the SWES knowledge base and the convenience of their technological use by the SWES inference engine. There is no need to describe the structures of the SWES knowledge base TBox area storages in details (Fig.3.), because they are described in the previous papers [7] [10]. In turn, talking about facts, which are stored in the specially designed storage, it is necessary to clarify that they are expressed in terms of RDF (Resource Description Framework).

Ontology is a key resource for the functioning of the SWES, because ontology is a source of knowledge for the SWES knowledge base. Before extracting knowledge from OWL ontologies, the SWES user prints the request, based on which the corresponding ontologies are searched in the Web and are supplied to OWL repository, which is part of the SWES knowledge base (Fig. 3.). After that found OWL ontologies, which are situated in the OWL repository, are retrieved and are merged into one OWL ontology. This merged OWL ontology is stored in a specially designated place of the knowledge base (Fig. 3) and serves as a source for rule generation. SWES is in need of fuzzy rules, and for this purpose it is necessary to store fuzzy values. It is logical to utilize OWL auxiliary means as comments, but this manner has some limitations what is mentioned in the previous section. The main limitation of storing fuzzy values in the comments of OWL ontology is that comments can be assigned only to the following OWL elements [11]:

- Classes,
- Properties,
- Individuals,
- Ontology headers.

OWL ontology elements, which cannot have comments, but they should have them for fuzzy rule generation, are the following [4] [7]:

- SubClassOf (partOf relation),
- EquivalentClass (equivalentOf relation),
- ComplementOf,
- ObjectProperty.

One of the principal features, which make it possible to eliminate the restriction of commenting the mentioned above OWL elements, is the ability to assign several comments to one OWL ontology element. By choosing a central element, which allows having comments, it is possible to develop a system of notation for the fuzzy values of the other elements of the OWL ontology. Certainly, classes are the most reasonable OWL ontology elements for this purpose among classes, properties, individuals and ontology headers, because classes satisfy by their level of abstraction and also by convenience in practical use, as will be shown hereinafter. Consider a class that has a few fuzzy relations (Fig. 4).

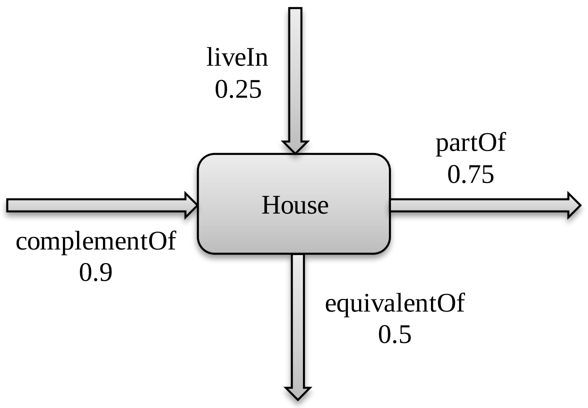

Fig. 4. Example of class with four relations.

In Fig. 4, the class “House” and “liveIn”, “partOf”, “complementOf”, “equivalentOf” relations are presented. The mentioned relations have corresponding fuzzy values of membership function and can be divided into two groups: group of incoming relations as “liveIn”, “complementOf” relations and group of outgoing relations as “partOf” and “equivalentOf” relations. There is a significant difference between incoming and outgoing relations. A class is a subject for outgoing relations, and it is an object for incoming relations. Using the example in Fig. 4, the class “House” is a subject for “partOf” and “equivalentOf” relations, while the same class is an object for “liveIn” and “complementOf” relations. The class affects neighboring classes by its outgoing relations, when it is a subject, and it is affected by other classes by dint of incoming relations, when it is an object. In this regard, it is possible to be assumed that outgoing relations in some way belong to the class and therefore only the values of membership functions of outgoing relations have to be stored in the comments of this class. It should be added that the values of membership functions of incoming relations have to be stored in the comments of the classes, which are the subjects for these relations. The assumption of incoming and outgoing relation division and their values of membership function storing in different places is a fundamental moment that gives an opportunity to develop strict storage system in order to store fuzzy values of membership functions for those OWL elements, which cannot have comments. Before describing the coding system of values of membership functions in the comments of classes, it is important to note that one comment is used for coding of the value of membership function of only one relation. Such a comment should have information about the type of relation, effect recipient that is the class, to which relation is connected and the actual value of the membership function. Comma can serve as a separator between the type of relation, effect recipient and the value of membership function. So, the format of a comment looks like as follows:

(<type of relation>,<effect recipient>,<value>).

There are only four types of relations, which have to be commented in such a way. These types of relations must have their own acronyms to be able to distinguish them. Table I contains the types of relations.

TABLE I. TYPES OF RELATIONS

Nr	OWL element	Type of relation	Clarification
1	subClassOf	partof	part_of relation between classes
2	equivalentClass	equivalentof	classes are equivalent
3	complementOf	not	classes are complement
4	ObjectProperty	link	arbitrary relation between classes

For example, if there are “House” and “City” classes, and the “House” class is a subclass of the “City” class, and the part_of relation between these classes has the value of membership function, which equals 0.75, then the comment will look like as follows:

“partof,City,0.75”.

In pursuance of the SWES information flow, which is shown in Fig. 1, rules and facts have to be adjusted for fuzzy inference after ontology. Rules and facts (entered by a user and produced by an inference engine) are closely linked with each other, and this means that change in one result in a change of the other. The Jena framework has an inference subsystem, which is designed to allow a range of inference engines or reasoners. There are a number of predefined reasoners [13]:

- Transitive reasoner;
- RDFS reasoner;
- OWL reasoner
- General purpose rule engine.

The transitive reasoner provides support for storing and traversing class and property lattices. The RDFS reasoner implements a configurable subset of the RDFS entailments. The OWL reasoner includes a default OWL reasoner and two smaller/faster configurations. Each of the configurations is intended to be a sound implementation of a subset of OWL Full semantics but none of them is complete in the technical sense. The general purpose rule reasoner supports user defined rules. It provides forward chaining, backward chaining and a hybrid execution model. Comparing the existing Jena reasoners, it is possible to conclude that the general purpose rule reasoner is the most suitable reasoner for use in the SWES, because it gives an opportunity to use custom rules what is vital for SWES functioning. The Jena rule syntax in abstract form looks like as follows [13]:

```
[DescriptionOrNameOfRule:
(condition to be met)
(another condition)
->
(fact to assert)
(another fact to assert)]
```

As it is clear from the Jena rule syntax the main profit from the application of rule is a new fact to be asserted if certain conditions are met. The ability of asserting or adding of new facts is useful from the SWES point of view but this is not enough for the realization of fuzzy reasoning. It is necessary to have a mechanism for selectively triggering rules, depending on the values of membership functions that is if these values are greater than certain value, then the rule is executed. There can help procedural primitives of the Jena inference subsystem [13]. The procedural primitives can be called by the rules, and they can optionally be used in the rule body, the rule head or both. But if the procedural primitives are used in the rule body then the primitive can act as a test - if it returns false the rule will not match. Primitives using in the rule head are only used for their side effects [13]. There are a lot of standard procedural primitives [13], but only several of them are necessary for fuzzy inference. In this connection Table 2 shows the procedural primitives of the Jena that may be needed to implement the fuzzy inference.

TABLE II. NEEDED PRIMITIVES.

Nr	Primitive	Operation
1	le(?x, ?y)	Test if x <= y
2	ge(?x, ?y)	Test if x >= y
3	sum(?a, ?b, ?c)	Sets c to be (a+b)
4	product(?a, ?b, ?c)	Sets c to be (a*b)
5	quotient(?a, ?b, ?c)	Sets c to be (a/b)

As to the facts, produced by an inference engine, they do not have to store the values of membership functions in the same format as it is described for OWL ontology elements, and the same applies to the facts specified by the user. The facts specified by the user have to store an initial value of membership function, but the facts, which are produced by an inference engine, have to store calculated values of membership functions.

It is necessary to describe in detail a working example of Jena inference, adapted for fuzzy reasoning, in order to understand the whole process from the beginning to end. Let us assume that there is an ontology, which consists of 2 classes “Car” and “Plane”. The “Car” class has two properties: engine and wheel. The “Plane” class has two properties, too. They are engine and wings as it is seen from Fig. 5.

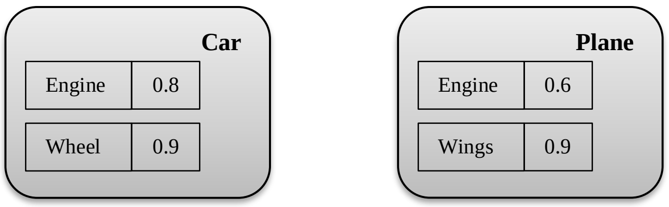

Fig. 5. Ontology of two classes.

In Fig. 5, it is seen that “Engine” property of the “Car” class has the value of membership function, which equals to 0.8 and “Wheel” property of the same class has the value of membership function, which equals to 0.9. In turn, “Engine” property of the “Plane” class has the value of membership function, which equals to 0.6 and “Wings” property of the same class has the value of membership function, which equals to 0.9. Now, based on [4] and [7] it is possible to generate the following rules:

IF Engine (0.8) **AND** Wheel (0.9) **THEN** Car (MIN[0.8,0.9]) (1)

IF Engine (0.6) **AND** Wings (0.9) **THEN** Plane (MIN[0.6,0.9]) (2)

The first rule means that if there is some object, which has a property “Engine” with the value of membership function, which is greater or equals to 0.8, and a property “Wheel” with the value of membership function, which is greater or equals to 0.9, then this object is a “Car” with the value of membership function, which equals to the minimum value of 0.8 and 0.9. Similarly, the second rule is understood. Further the generated rules have to be transformed to the rules in the format of the Jena rule format to have an opportunity of reasoning on facts. The rules in the format of the Jena format look like as follows:

```
@prefix rdf:http://www.w3.org/1999/02/22-rdf-
@prefix ex: http://example.com/#

[Car:       (?s rdf:type ex:Engine)
            (?s ex:fuzzy ?pw)
            ge(?pw,0.8)
            (?z rdf:type ex:Wheel)
            (?z ex:fuzzy ?px)
            ge(?px,0.9)
            min(?pw,?px,?u)
            strConcat('Car',' ','vmf=',?u,?a)
            ->
            (ex:NF rdfs:Comment ?a)
]

[Plan       (?r rdf:type ex:Engine)
            (?r ex:fuzzy ?w)
            ge(?w,0.6)
            (?y rdf:type ex:Wings)
            (?y ex:fuzzy ?x)
            ge(?x,0.9)
            min(?w,?x,?t)
            strConcat('Plane',' ','vmf=',?t,?b)
            ->
            (ex:NF rdfs:Comment ?b)
]
```

Now let us suppose that the user typed into the search bar of an expert system the following data with the values of membership functions (Fig. 6):

engine (0.85) wheel (0.9) wings (0.1)

Fig. 6. The request of the user to an expert system.

After receiving a user's request, the expert system constructs a data file, which has RDF format. The Jena utilizes the general purpose rule engine for this data file and obtains the result. The necessary result is concentrated in some class as a comment. In our case the needed result of reasoning is the following:

"Car vmf=0.85".

Analyzing the result, it should be noticed that "Car" is an inferred assertion and "vmf=0.85" is the value of membership function, which equals to 0.85. It is expected that new rules will produce new facts, which will be placed in the same class in the form of a comment. This manner of new fact placing allows accessing the new facts only that greatly simplifies the output results to the user.

IV. CONCLUSION

This paper is dedicated to the problem of adaptation of the Jena framework for fuzzy reasoning. Such an adaptation is necessary in order to have an opportunity to integrate the Jena framework with the Semantic Web Expert System. The Jena framework, which is adapted for fuzzy reasoning, is aimed at reasoning the fuzzy rules, which are generated in the process of OWL ontology merging [7].

In the paper the requirements for the purpose of the Jena adaptation for fuzzy reasoning in the Semantic Web Expert System are worked out. The flow of information in SWES, which runs from the OWL ontology to the new facts through rules, is presented and then the requirements for each part of this flow are defined. After that, the realization of the Jena, adapted for fuzzy reasoning, is described. In the process of this realization the SWES knowledge base is supplemented by new storage for placing facts. The system of values of membership functions storing in OWL ontology was also developed. The Jena framework has a powerful inference subsystem, which is expressed as a set of several standard reasoners. These reasoners were overviewed, and one of them was soundly selected. And finally, the working example of the Jena framework, which is adapted for fuzzy reasoning, was described.

The task of adaptation of the Jena framework for fuzzy reasoning is necessary for implementation in the Semantic Web Expert System; however such an adapted reasoning system can be useful in other projects, where fuzzy reasoning is necessary and where the Jena framework is chosen. Certainly, it will be valid as long as the Jena developers do not work out the built-in ability of fuzzy reasoning.

REFERENCES

[1] J. Davis, R. Studer, P. Warren, "Semantic Web Technologies Trends and Research in On-tology-based Systems," John Wiley & Sons Ltd, Chichester, 2006

[2] T. R. Gruber, "A translation approach to portable ontologies," *Knowledge Acquisition*, 5(2):199-220, 1993

[3] W3C OWL Working Group, "OWL2 Web Ontology Language Document Overview (Second Edition)," Available online: http://www.w3.org/TR/owl2-overview/

[4] O. Verhodubs, J. Grundspeņķis, "Evolution of Ontology Potential for Generation of Rules," Proceedings of the 2nd International Conference on Web Intelligence, Mining and Semantics, Craiova, 2012

[5] O. Verhodubs, J. Grundspeņķis, "Towards the Semantic Web Expert System," RTU Press, Riga, 2011

[6] P. Jackson, "Introduction to Expert Systems, Third Edition," Addison-Wesley, 1998

[7] O. Verhodubs, J. Grundspenkis, "Ontology merging in the context of a Semantic Web Expert System," Springer, Saint-Petersburg, 2013

[8] O. Verhodubs, J. Grundspenkis, "Comparison of ontology reasoning systems for SWES," in proceedings

[9] F. Bobillo, U. Straccia, "Fuzzy Ontology Representation using OWL 2," International Journal of Approximate Reasoning, vol. 52, pp. 1073–1094, 2011

[10] O. Verhodubs, J. Grundspenkis, "Algorithm of ontology transformation to rules," Riga : RTU Press, 2013

[11] W3C, "OWL Web Ontology Language Reference," Available online: http://www.w3.org/TR/owl-ref/

[12] Apache Software Foundation, "Jena API – Common ontology application problems," Available online: http://jena.sourceforge.net/ontology/common-problems.html

[13] Apache Software Foundation, "Reasoners and rule engines: Jena inference support," Available online: http://jena.apache.org/documentation/inference/index.html