Apogee: Application Ontology Generation from Domain Ontologies

Iaakov Exman and Danil Iskusnov

Software Engineering Department, The Jerusalem College of Engineering - Azrieli
POB 3566, Jerusalem, 91035, Israel

Abstract. To achieve runnable knowledge from the highest abstraction level of an application one needs to start with a set of application ontologies. But application ontologies are not readily available in the literature. One needs to generate dedicated and smaller application ontologies from larger generic domain ontologies. The main problem to be solved is to generate application ontologies with optimal size given two opposing criteria: to enlarge the selected domain ontology segments to include most relationships between relevant concepts, as opposed to reduce the same segments to exclude irrelevant terms. The proposed solution is to traverse the domain ontology just one level upward from the last relevant keywords found. The work also describes a chain of algorithms and practical techniques to reach the proposed solution stage. Finally, case studies are used to actually illustrate the whole approach.

Keywords. Application ontology, Domain ontology, Keyword extraction, Classification, Optimal Application Ontology Size.

1 Introduction

Recently [8], [18] we have been extending MDE (Model Driven Engineering), which starts software system development from a UML design model, to KDE (Knowledge Driven Engineering) in which one starts from one level of abstraction higher, viz. from a set of ontologies. This set of ontologies is what we call application ontologies.

Application ontologies are dedicated to a chosen software system that one wishes to develop. Thus they are much smaller than the generic domain ontologies, whose set of concepts contain the respective application ontologies' concepts as sub-sets.

The main issue of this paper is a systematic procedure to obtain the desired set of application ontologies, given the generic domain ontologies. Our approach is to combine a chain of existing algorithms within an integrated tool – coined Apogee – for generation of specific application ontologies.

A central stage of the referred algorithm chain is to decide the optimal size of the resulting application ontology, such that it may still be considered a legitimate ontology – including most of the relationships among relevant keywords – while excluding as far as possible terms irrelevant to the specific application.

Exman I. and Iskusnov D..
Apogee: Application Ontology Generation from Domain Ontologies.
DOI: 10.5220/0005181000310042
In *Proceedings of the 5th International Workshop on Software Knowledge* (SKY-2014), pages 31-42
ISBN: 978-989-758-051-2

1.1 Related Work

Literature reviews concerning ontologies are available in refs. Guarino [9],[10] and Nguyen [13].

There exist some types of generic ontologies as mentioned in the next section of this paper, and specific types of ontologies for various purposes. Among the latter, one finds reputation ontologies (e.g. Chang et al. [1]) and opinion ontologies (e.g. Exman [7]).

Tools of interest for operations involving ontologies include among others, Ontobraker (Decker et al. [4]) for ontology based accessing distributed information, Swoogle (Ding et al. [5]) a search engine for the semantic web and ontologies, and Protégé (from Stanford University [14]) for ontology editing and other operations.

A sample of application oriented papers are Jean-Mary et al. [11] for ontology matching with semantic verification, Luke et al. [12] about ontology-based web-agents, and Swab-Zamazal [16] about the influence of pattern-based ontology refactoring on ontology matching.

A project having some similarity to ours is ReDSeeDS, viz. Requirements-Driven Software Development System [15]

The remaining of the paper is organized as follows. We describe types and sizes of ontologies (section 2), introduce the goal of our work (section 3), outline the solution chain of algorithms in the overall process from domain to application ontologies (section 4), overview the software architecture and implementation of our Apogee tool (section 5), present case studies as an illustration of the approach (section 6) and conclude with a discussion (section 7).

2 Ontology Types and Sizes

One can roughly classify ontologies into specific and generic types. Specific types of ontologies have a narrowly defined purpose, and a relatively small size, such as reputation ontologies (e.g. [1]). Generic types can refer to whole domains, and diverse applications, and are mostly of large size.

In this paper we continue our investigation of special purpose, small ontologies. In particular, we deal with the extraction of specific/small ontologies from generic/large ones.

2.1 Domain Ontologies

A domain ontology is characterized by being comprehensive, i.e. to encompass all possible concepts that may appear in a given domain of discourse. Domain ontologies are usually of large sizes and may have a complicated structure, containing generic concepts, their sub-types, instances and properties.

2.2 Application Ontologies

Application ontologies are a notion proposed in our work. We assume that a given software system or application is defined in its highest abstraction level by a set of related application ontologies.

For instance, the development of a software application dealing with purchases in the internet – as was dealt with in ref. [18] – may be started from two ontologies: one defining the "*shopping cart*" and another one defining a "*product*".

Typical concepts in the *shopping cart* ontology are: products, items per product, tax, current price and total price. Common concepts in the *product* ontology can be: product name, product price, serial number and part number.

Concepts in an application ontology are either candidates for UML classes or for these classes' attributes.

3 The Goal: From Domain to Application Ontologies

In this section we motivate the need for systematic extraction procedures of smaller ontologies dedicated to a desired application from generic domain ontologies.

3.1 KDE: Knowledge Driven Engineering

In our recent approach (see e.g. [8]) to develop software systems from a higher abstraction level than UML, i.e. starting from a set of application ontologies, these ontologies serve to provide semantics for scenarios describing possible behaviors of the system.

The application ontologies have been formulated manually a priori, and were gradually refined as needed, when new scenarios were added to describe the system more precisely.

For instance, in the internet purchase example, we have used an application ontology for the shopping cart as seen in Fig.1.

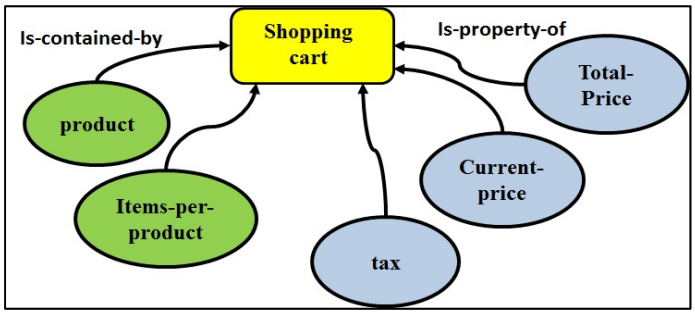

Fig. 1. Sample shopping-cart ontology for an Internet Purchase – this ontology was formulated manually ad hoc. In the l.h.s. of the ontology {product, items-per-product} have the *Is-contained-by* relationship. In the r.h.s. {Total-price, Current-price, tax} have the *Is-property-of* relationship to Shopping-cart.

A typical scenario for the same internet purchase example is shown in Fig. 2.

Internet Purchase – Sample Scenario
Pick a shopping-cart;
Put one item of product A in the shopping-cart;
Put three items of product B in the shopping-cart;
Proceed to payment.

Fig. 2. Sample Scenario for an Internet Purchase – Note that the terms {product, items-per-product, shopping-cart} appear in the shopping-cart ontology, but the term {payment} does not appear in the ontology.

Note the lack of correspondence of terms in the ontology to terms in the scenario. Some of the terms in the ontology do not appear in the scenario and vice-versa.

3.2 The Problems

Problems with the manual ad hoc formulation of application ontologies include among others:

- *Lack of Systematic Approach* – no guarantee of consistency within an application ontology and along time;
- *Lack of Knowledge Reusability* – formulation needs to be done anew even for similar systems in the same domain.

Thus, we are led to seek a systematic procedure to obtain application ontologies from domain ontologies.

4 The Solution: From Keywords to an Optimal Size Ontology

In this section we propose our solution, on how to deal with the process leading from generic domain ontologies to smaller ontologies dedicated to a desired application. We describe the chain of relevant algorithms, starting from the inputs to the process and emphasizing the main issue of attaining an optimal size for the application ontology.

The overall idea is that a software stakeholder or the software developer knows at least the desired behavior of the software system. This behavior is expressed in a text, say a scenario like that in Fig. 2. From this knowledge one will extract keywords, to characterize the suitable domain ontologies. Then we get into techniques to reduce the latter into smaller application ontologies.

4.1 Inputs

The inputs to the process are:

- *bank of ontologies* – a bank of potentially useful domain ontologies;
- *text* – typically user provided scenarios that should be executable by the application.

4.2 The Chain of Relevant Algorithms

The proposed chain of algorithms relevant to the generation of application ontologies from domain ontologies is as follows:

- *Keyword Extraction* – from the input text (scenario) select relevant application keywords. For the internet purchase example these could be: shopping-cart, item, product, payment.
- *Domain Classification* – assign keywords to a domain, in order to select potential domain ontologies from the bank of ontologies. For instance 'payment' belongs to the 'financial' domain. An important issue here is to resolve ambiguities: does 'bridge' refer to card games or to civil engineering structures?
- *Matching Ontology Segments* – perform matching of the keywords in the keyword list to selected domain ontologies' and extract segments of those ontologies including the matched keywords. Here resides the main problem concerning the ***optimal size*** of the application ontology. This will be explained in detail in sub-section 4.3.
- *Compose New Application Ontology* – from matching ontology segments;
- *Check Composed Application Ontology* – for consistency and relationships subsumption among classes.

4.3 The Optimal Size of the Application Ontology

Here we focus on the main issue of optimal size of the application ontology, which occurs in the *matching ontology segments* stage. We first explain the problem, then propose our specific solution.

The matching of ontology segments starts by searching, for all keywords, all the appearances of a given keyword in the relevant domain ontology. For each such appearance, one traverses the domain ontology graph upwards, until one either gets to the root (say "thing") or there is a criterion which tells one to stop earlier. All the terms before stopping are included in the current ontology segment.

The referred criteria are the significant part of the solution. An example of criterion is "one encounters a term in a previously found segment", thus one can coalesce the new segment with the previous one.

Our solution proposes the following criterion. The upward traversing stops "one level above the last term relevant to the set of keywords". The specific meaning of relevance should be formally further specified.

This last criterion strives to include all the relevant terms and relationships, while excluding irrelevant terms, to achieve an optimal size of the application ontology.

5 Apogee: Design and Implementation

Apogee is the tool that we have designed and implemented to generate dedicated application ontologies from larger generic domain ontologies. As far as possible, we have reused external tools by means of calls to their APIs.

5.1 Apogee Design

The modules of the Apogee tool are schematically shown in Fig. 3. It contains core internal modules and external tools called by means of APIs.

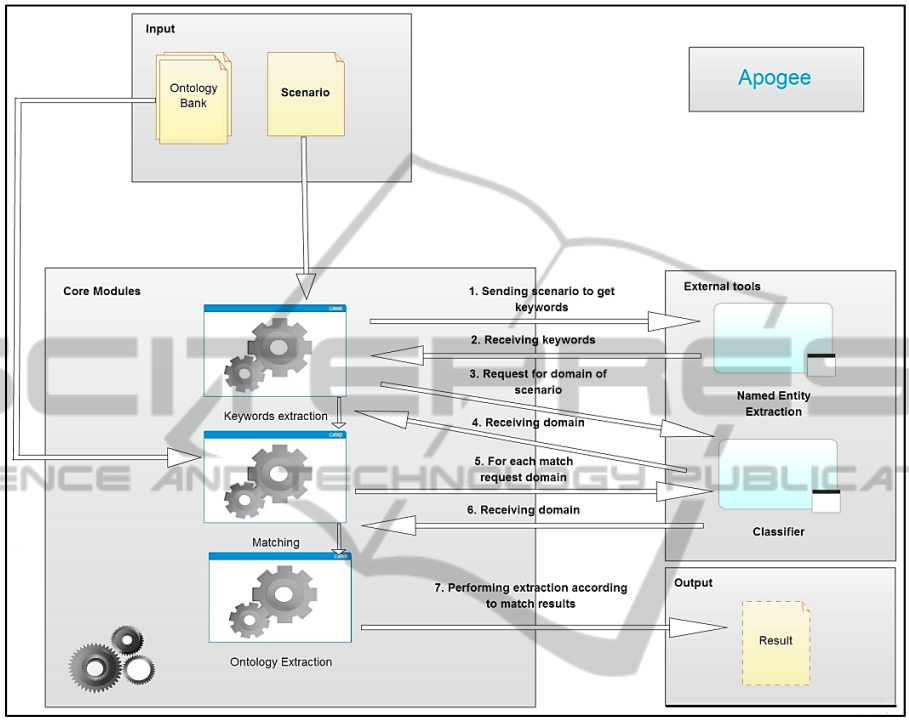

Fig. 3. *Apogee Schematic Design* – It shows: a- Input; b- External tools called by core modules; c- Core Apogee modules; d- Output. Arrows display the order of execution.

5.2 Apogee Implementation

Apogee core modules were implemented in Java. They use Java libraries such as:

- OWL API – for creating, manipulating and serializing OWL ontologies;

The external tools used were:

- dataTXT-NEX – as Named Entity Extraction [2];
- dataTXT-CL – as text classifier [3].

The output format is textual OWL.

6 Case Studies

We have made experiments with several case studies to validate the approach. These included diverse specific variations such as providing as input to the domain classifier the whole scenario, or sentence by sentence individually.

Here we show one of the case studies, the ATM cash withdrawal, to illustrate results and difficulties.

6.1 Case Description: ATM Cash Withdrawal

In Fig. 4 one can see a sample input text and selected keywords. It uses a Gherkin syntax in the easily applied and understood [**Given, When, Then**] format, used by the Cucumber [17] tool, to describe the scenario.

These scenarios can be input in various "natural" languages. Note that the syntax does not demand grammatically well-formed sentences.

ATM Cash Withdrawal – Sample Scenario

Account has sufficient funds.
Given the account balance is 100.
And the Card is valid.
And the ATM contains 50.
When the Account requests cash 20.
Then the account balance should be 80.
And the card should be returned.

Fig. 4. ATM Cash Withdrawal *Sample Input scenario and selected keywords* – The scenario uses the Gherkin [**Given, When, Then**] format. Selected keywords have a green background.

It is probably preferable to input the scenario as a whole to the domain classifier, instead of sentence by sentence. For instance, suppose that the second sentence is just:

Given the balance is 100.

In the context of the whole scenario, a human would certainly understand this sentence, assuming that balance refers to the account. But as an individual sentence, balance is ambiguous and could have several different meanings in various domains.

6.2 Domain Classifier: ATM Cash Withdrawal

The domain classifier outcome for the above ATM Cash Withdrawal scenario is seen in Fig. 5 below.

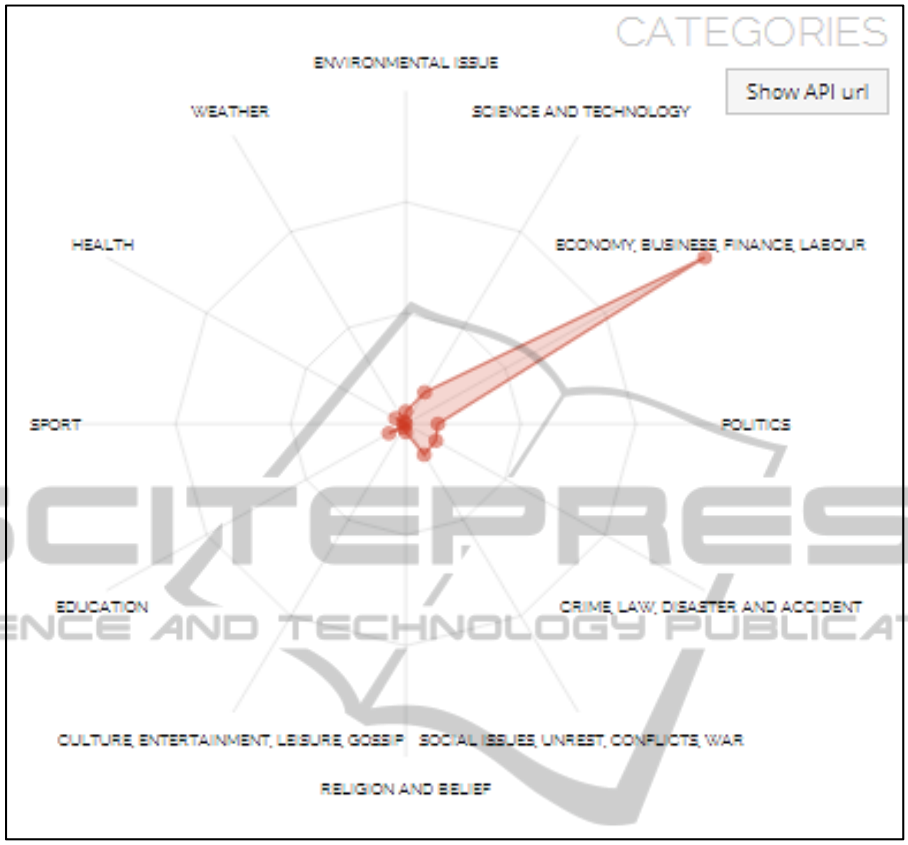

Fig. 5. ATM Cash Withdrawal *Classifier Outcome* – From the categories suggested by the classifier, the chosen category is "ECONOMY, BUSINESS, FINANCE, LABOUR" as shown by the biggest red pointer in the figure.

One should note that a keyword consisting of "*balance*" alone, without account, could give a completely different result. Balance of forces has a physical meaning, fitting "SCIENCE AND TECHNOLOGY". Balance can refer to balance of power among parties, fitting "POLITICS". Balance can be a quality of a basketball player or a characteristic of a judo competition, fitting "SPORT". Balance between preys and predators, fits "ENVIRONMENTAL ISSUE".

6.3 Application Ontology Results

The application ontology for the previous values of the classifier is shown in Fig. 6. Ontology segments were obtained by going upwards from the matched keyword. The textual OWL output was inserted into Protégé [14] for display purposes.

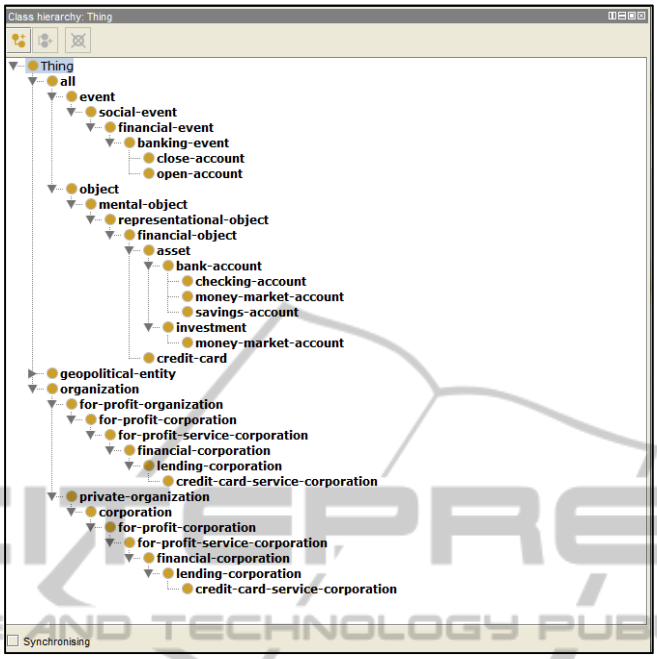

Fig. 6. ATM Cash Withdrawal *Application Ontology Classes* – This is the class hierarchy displayed by the Protégé tool.

Next in Fig. 7 one sees the object properties of the same application ontology.

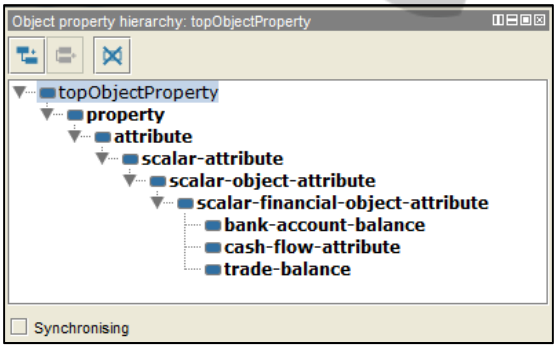

Fig. 7. ATM Cash Withdrawal *Application Ontology Object Properties* – This is the object property hierarchy displayed by the Protégé tool.

6.4 Application Ontology Analysis

The above application ontology results for the ATM Cash Withdrawal case study show both the quality and the difficulties of the outcome, enabling a better understanding of the optimal size problem concerning the application ontology.

On the positive side, one can see that relevant terms – such as bank-acount, close-

account, open-account, credit-card – and properties – such as bank-account-balance – indeed appear in the application ontology.

On the negative side, it is quite obvious that a lot of the included terms are either too generic – e.g. social event, above financial event – or irrelevant to the ATM Cash Withdrawal – e.g. for-profit-service-corporation, in fact all terms containing 'corporation'.

These considerations show that going upwards from say 'bank account' to the root class "thing" is far from optimal. One can see that besides the abstract criterion to stop "one level above the last term relevant to the set of keywords", which in the case of 'bank account' would be 'asset', one needs concrete formulas to apply the criterion in practice.

7 Discussion

This work proposes the generation of small specific application ontologies from much larger and comprehensive domain ontologies, in an almost fully automatic way. The approach has been actually implemented into the Apogee tool. The tool was tested by case studies. A sample case study was shown in this paper to illustrate the ideas and the approach.

7.1 Open Issues and Future Work

Several issues have been raised along this work.

Concerning the input composition and size, there is a trade-off. If one inputs a full scenario at once to Apogee, the many keywords facilitates resolution of domain ambiguity, on the other hand may introduce additional noise which is difficult to control. The alternative is to input the scenario line by line, in which case ambiguity is more prevalent, but one has less noise. The issue seems to be incompletely resolved.

Regarding the output, again one could decide a priori, by human intervention, the number of output application ontologies: one single larger application ontology or two smaller ones, etc. One would like to be able to formulate criteria to solve this issue in a more clear-cut way.

With respect to the main issue of optimal size of the application ontology, one needs quantitative formulas for the criteria on how to include/exclude terms in the vicinity of the selected keywords. The 'one level above the keywords' is ad-hoc and still needs deeper testing. Possible quantitative formulas to eliminate irrelevant terms include those found in data mining areas, say TfIdf and the notion of Relevance within the context of the interestingness approach [6]. This issue deserves extensive investigation.

The whole line of research about special purpose small or nano-ontologies, is still in need of a unified approach, in particular referring to their semantic significance and their "legitimate" status as first-class citizens of the ontology world.

An interesting open issue is the demands from application ontologies in order to enable their reuse for different systems having the same requirements and purpose.

The current Apogee is an initial prototype. A second version – after

comprehensive testing of the current version – could be much more flexible, e.g. to include APIs for various alternative external tools.

7.2 Main Contribution

The main contribution of this work is a systematic approach for generation of specific smaller application ontologies from much larger and comprehensive domain ontologies. A particular problem of interest is emphasized, viz. the optimal size of the application ontologies.

References

1. Chang, E., Hussain, F.K. and Dillon, T., "Reputation ontology for reputation systems", in SWWS International Workshop on Web Semantics, pp. 957-966, Springer-Verlag (2005). DOI: 10.1007/11575863_117.
2. dataTXT-NEX – dandelion named entity extraction API. Web site: https://dandelion.eu/products/datatxt/nex/demo/
3. dataTXT-CL – dandelion text classification API. Web site: https://dandelion.eu/products/datatxt/cl/demo/
4. Decker, S., Erdmann, M., Fensel, D. and Studer, R., "Ontobroker: Ontology based access to distributed and semi-structured information", In Proc. DS-8, pp. 351-369, (1999).
5. Ding, L., Finin, T., Joshi, A., Peng. Y., Cost, R.C., Sachs, J., Pan, R., Reddivari, P. and Doshi, V., "Swoogle: A Semantic Web Search and Metadata Engine", in Proc. CIKM'04, ACM (2004). Tool web site: http://swoogle.umbc.edu/
6. Exman, I., "Interestingness – A Unifying Paradigm – Bipolar Function Composition", in Fred, A. (ed.) Proc. KDIR'2009 Int. Conference on Knowledge Discovery and Information Retrieval, pp. 196-201, (2009).
7. Exman, I., "Opinion-Ontologies: Short and Sharp", accepted for publication in 6th KEOD, Int. Conf. on Knowledge Engineering and Ontology Development (2014).
8. Exman I. and Yagel R.: "ROM: An Approach to Self-Consistency Verification of a Runnable Ontology Model", in A. Fred, J.L.G. Dietz and J. Filipe (eds.) Knowledge Discovery, Knowledge Engineering and Knowledge Management, 4th International Joint Conference IC3K, Revised Selected Papers, CCIS, vol. 415, pp. 271-283, Springer Verlag, Berlin, Germany (2012).
9. Guarino, N. "Semantic Matching: Formal Ontological Distinctions for Information Organization, Extraction and Integration", Summer School on Information Extraction, Frascati, Italy, in LNCS Vol. 1299, pp. 139-170, Springer-Verlag (1997).
10. Guarino, N. "Formal ontology in information systems", in Guarino, N., (ed.), Formal Ontology in Information Systems, pages 3–18, Amsterdam. IOS Press, (1998).
11. Jean-Mary, Y. R., Shironoshita, E.P. and Kabuka, M.R., "Ontology Matching with Semantic Verification", Web. Semant. Vol. 7 (3), pp. 235-251, (September 2009). DOI: 10.1016/j.websem.2009.04.001.
12. Luke, S., Spector, L., Rager, D. and Hendler, J., "Ontology-based web agents", In Proc. AA97 1st Int. Conf. on Autonomous Agents, pp. 59-66, (1997).
13. Nguyen, V., "Ontologies and Information Systems: A Literature Survey", Technical Report DSTO-TN-1002, Australian Government, Defence Science and Technology Organization, (2011). Web site: http://digext6.defence.gov.au/dspace/bitstream/1947/10144/1/DSTO-TN-1002 PR.pdf

14. Protégé - A free, open-source ontology editor and framework for building intelligent systems. Web site: http://protege.stanford.edu/.
15. ReDSeeDS – Requirements-Driven Software Development System. Web site: http://smog.iem.pw.edu.pl/redseeds/ (2014).
16. Svab-Zamazal, O., Svatek, V., Meilicker, C. and Stuckenschmidt, H., "Testing the Impact of Pattern-Based Ontology Refactoring on Ontology Matching Results", in Proc. Ontology Matchin Workshop, hosted by ISWC 2008, 7th Int. Semantic Web Conference, pp. 229-233, (2008).
17. Wynne, M. and Hellesoy, A.: *The Cucumber Book: Behaviour Driven Development for Testers and Developers,* Pragmatic Programmer, New York, USA, (2012).
18. Yagel, R., Litovka, A. and Exman I.: *KoDEgen: A Knowledge Driven Engineering Code Generating Tool*, in Proc. 4th SKY'2013 International Workshop on Software Knowledge, hosted by IC3K, pp. 24-33, Vilamoura, Portugal, (2013).