Ontological reusability in state-of-the-art semantic languages

Michał SZYMCZAK, Maciej GAWINECKI,
Mladenka VUKMIROVIC, Marcin PAPRZYCKI

Abstract: Evolving reusable domain ontologies and information described using these ontologies are the key aspects of the Semantic Web. In this paper, we use an example of ontology merging to discuss ontology reusability in the context of existing semantic languages and software tools for ontology management.

1. Introduction

Recently, we have proposed utilization of semantically demarcated data in agent-based Travel Support System (TSS) [22, 24] and have developed ontologies of a *hotel* and a *restaurant* [20, 21, 23]. Currently we are extending these ideas to an agent-based airline ticket auctioning system [38] and proposed an initial version of a generalized air travel ontology [39]. Development of these three ontologies proceeded almost without communication between their authors. The *hotel ontology* was based on analysis of hotel descriptions stored in Internet-based travel agencies. The *restaurant ontology* resulted from the RDF demarcated data stored in the ChefMoz repository. Finally, the *air travel ontology* originated from in-depth study of the *International Air Transport Association* (IATA) manuals and information available in Global Distribution Systems (GDS) [31, 32, 33, 37]. Note that this situation is very similar to the development of the Semantic Web [11], where different teams create more or less comprehensive (sub)domain ontologies that are one day to be merged (or interact with each other). Obviously, hotels, restaurants and air travel are closely interrelated as pars of "world of travel" and thus we decided to merge their ontologies. Initially, we have shallowly combined ontologies of hotel and restaurant and the results were promising [21]. However, since the TSS has to deal with "all" possible travel related objects, we had to start investigating issues involved in robustly combining travel ontologies. Therefore, we had to start facing issues related to: (a) differences between combined hotel and restaurant ontology and the air travel ontology, (b) size of the resulting ontology, (c) necessary preparations to include other travel objects, such as: railroad, cinema, opera etc., (d) robustness and flexibility of the combined ontology that has to be prepared for further domain extensions, upgrades and modifications. Since one of the mantras of the Semantic Web is reusability of existing ontologies, when starting our work, we have tried to perform a simple integration of our system with a currency ontology defined in an existing agent-based currency exchange system – Cambia service [6]. Problems we have run into indicated that there exists a whole set of issues that are related incompatibilities between semantic language and tools that are to operate on them.

The aim of this note is to discuss these issues. We proceed as follows. First, we briefly examine existing semantic languages and elaborate the language choice made for our system. Next we discuss issues involved in combining hotel, restaurant and air travel ontologies. Lastly an attempt to integrate external *Cambia* agent system with our system and problems which have occurred is described and analyzed.

2. Why Resource Description Framework?

While a number of languages for semantic content demarcation have been created, three of them are most popular today:

- *Extensible Markup Language* (XML) and *XML Schema* (XSD),
- *Resource Description Framework* (RDF) and *RDF Schema* (RDFS),
- *Web Ontology Language* (OWL).

Creating general travel domain ontology for the TSS brought us to define the following requirements for the ontology language:

1. machine and human readability;
2. support for inference rules, particularly support for *"is-a"* relationship, i.e.:
    a. *top-down inheritance* of attributes,
    b. *bottom-up inheritance* of instances;
3. means for reuse of existing ontologies.

In other words, such language should allow agents and humans to: (a) interpret resources as a representative of particular classes, (b) define new classes and their properties, (c) attach human readable labels, (d) organize classes in hierarchical order and (e) reuse already defined concepts. Note that in our context precision of a language expression is not of key importance. Let us now compare the three languages.

2.1. RDF and RDFS vs. XML and XSD

XML provides an easy formal syntax for document description, while XSD allows defining data types and data structures [35, 12, 13, 14]. In Table 1 we present main differences between RDF(S) and XML with XSD:

RDF(S)	**XML with XSD**
Only basic data types from XSD	Data types and structures
Ordering elements must be explicitly defined with collection constructs; otherwise it does not matter	Order of elements does matter, their order is part of semantics
Semantics based on RDF graphs	Semantics based on XML infosets [26]
Types of elements are classes	Types of elements do not need to have any meaning
Supports *"is-a"* relationship	Lacks support for *"is-a"* relationship

Table 1 : Main differences between RDF(S) and XML with XSD [18, 19, 25, 35]

Please note that although RDF is an application of XML, it delivers triple-based semantics. The role of each element in a statement is precisely defined and allows for inference on described resources. Moreover, one can define range and domain of each property in RDF(S). The latter feature enables defining constraints on properties and delivers additional inference rules for both, object and subject, of the property. Pure XML does not provide such rich semantics. Moreover, structure of an XML document cannot vary from its XSD definition, while document structure flexibility is one of the features of RDF(S). Thus XML with XSD was found unsuitable for our system.

2.2. OWL vs. RDF(S)

OWL is based on RDF and introduces the following additional features [35, 16, 17]:

- functional and inverse functional properties;
- local domain and range constraints;
- cardinality constraints;
- transitive and symmetric properties (e.g. *sameAs* property);
- logic-based class construction (union, intersection, complement, restriction);
- one URI for whole ontology.

Both, OWL and RDF meet our requirements. Moreover, OWL not only has all features of RDF, but also brings richer vocabulary which improves inference and precision of resource description. The reason for not using OWL is that we did not need all its features, such as setting constraints and cardinalities on classes and assigning a separate URI for each ontology. Neither the explicit declaration of external ontology import nor similarity between concepts definition are required. Despite the fact that RDF is less precise, it seems to provide greater reusability of ontologies, as they are more often expected to simply conceptualize particular domain rather than express complex constraints of concepts. What is also important is better performance of RDF reasoners and parsers in comparison with similar OWL tools [18, 25].

3. Reuse of RDF ontologies

Originally, we created two autonomous ontologies of: hotel, restaurant. In [21] we have presented elementary integration of hotel and restaurant ontologies. This involved only simple reference points, such as for instance named properties for stating that a restaurant can be placed inside of a hotel or that a particular restaurant and a particular hotel are placed close to each other. Recently co-development of both ontologies became more mature, air travel ontology has been introduced and some sub-ontologies have been distinguished. These modifications forced us to re-consider integration of all ontologies. Such integration proceeds iteratively and consists of following stages:

1. identification of intersections between ontologies that are to be integrated,
2. merging discovered common parts,
3. updating references to other concepts.

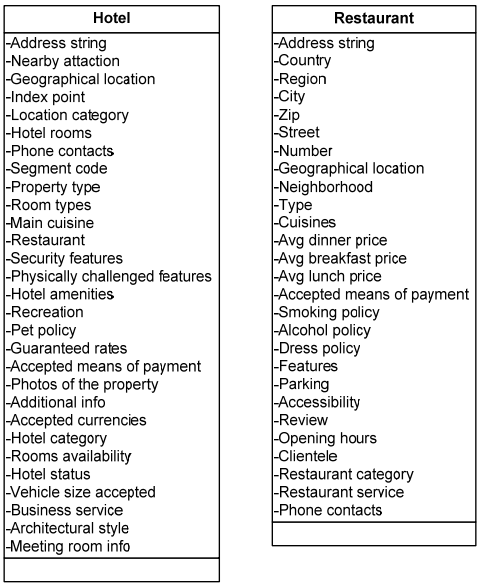

Figure 1. Hotel and Restaurant ontologies before location property integration

3.1. Location description unification

Geographical location is one of common features of restaurant and hotel ontologies. Figure 1 presents both ontologies before integration. Each class describes geographical location through its own set of geographical properties, such as: street address, country, city/town, region, zip code, reference points or location description. Consequently, a separate class called *OutdoorLocation* can be created and union of location describing properties from both classes assigned to this class.

After separation of concepts, we proceed with integration. As both, *Hotel* and *Restaurant*, classes are temporarily deprived of any geographical location description they can become a sub-classes of *OutdoorLocation*. Due to the definition of sub-class relationship both classes inherit properties which are defined for their parent classes. Moreover, instances of a hotel or a restaurant can be perceived as instances of *OutdoorLocation* class. This generalization, when implemented in RDF, does not force us to define all geographical location properties when an instance of a hotel or a restaurant is created. If, for example, geo-position is unknown and we want to fill only address properties then *geoPosition* property can be omitted. The resulting, combined ontology can be found in Figure 2.

Additional advantage of the proposed hierarchy is that it allows creating new classes which represent particular geographical location in a simplified manner. It is perfectly enough to create such a class as a sub-class of *OutdoorLocation* and define the class-specific properties. This issue appeared when we wanted to develop airport location description for the purpose of the air travel ontology (see Figure 3).

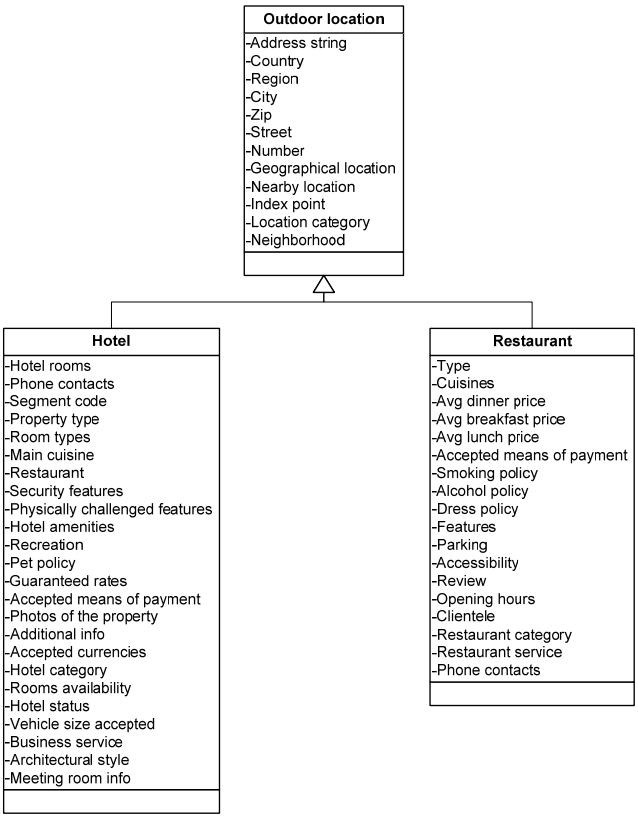

Figure 2. Hotel and Restaurants – subclasses of Outdoor Location

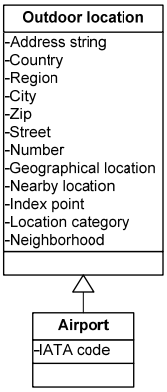

Figure 3. Airport – subclass of Outdoor Location

3.2. Sharing concept of a discount

Let us now consider similarity between hotel, restaurant and air travel ontologies found in the discount concept. All three ontologies use it; containing the same information:

- code of the particular discount,
- amount of reduction of the base-price,
- short description of the discount policy.

However, IATA defined specific air travel discount codes. The question has arisen: how to integrate these with hotel and restaurant discount codes (including both OpenTravel Alliance (OTA) [34] specific and general discounts – omitted in the OTA specification). For the purpose of integration we have distinguished specific discount codes and defined those as sub classes of general *DiscountCodes* class (see Figure 4).

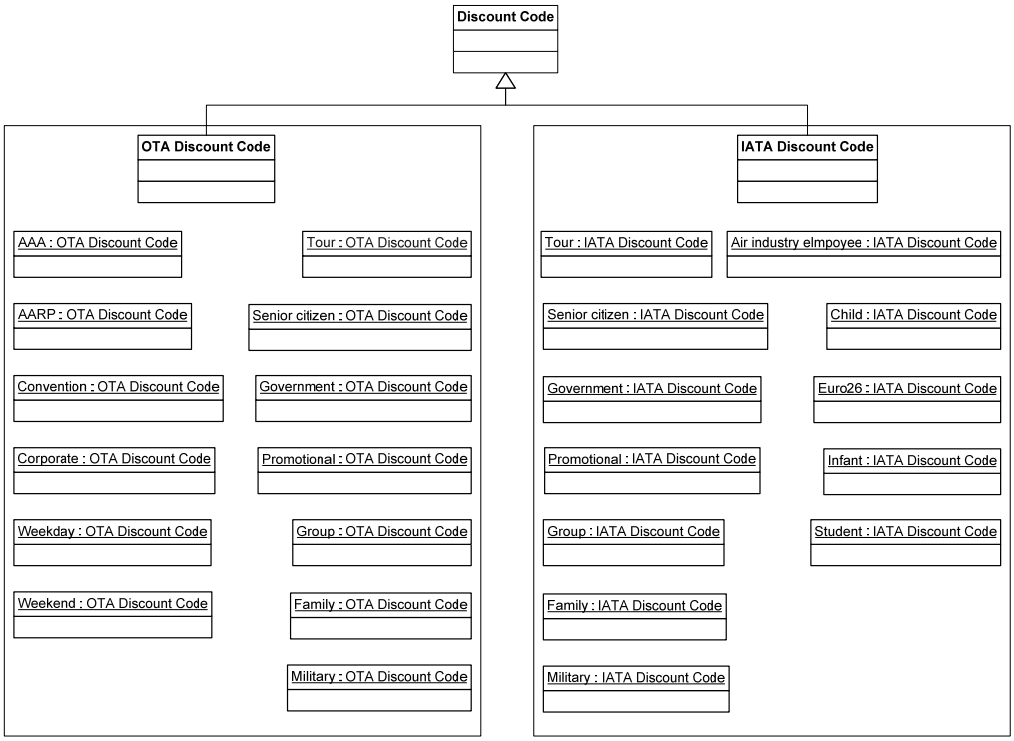

Figure 4. Discounts class and initial hierarchy of Discount Codes

Furthermore, *Family*, *Government*, *Group*, *Military*, *Promotional*, *Senior Citizen* and *Tour Conductor* discount codes appear in both IATA and OTA specifications. This allows us to proceed with a tighter integration. We can distinguish a class of common discounts – *IataOtaCommonDiscountCodes* – and define the seven codes as its instances. Defining common discount codes as a sub-class of *OTADiscountCodes* and

IATADiscountCodes led us to specification of concept equivalence between OTA and IATA specifications, again, without making any explicit equivalence declaration. The final result of integration is depicted in Figure 5. Common codes are identified as members of the same class. The sub-class relation defines them also as instances of both: OTA and IATA discount codes classes. Hence, a group of codes can be used either in hotel, restaurant or air ticketing system.

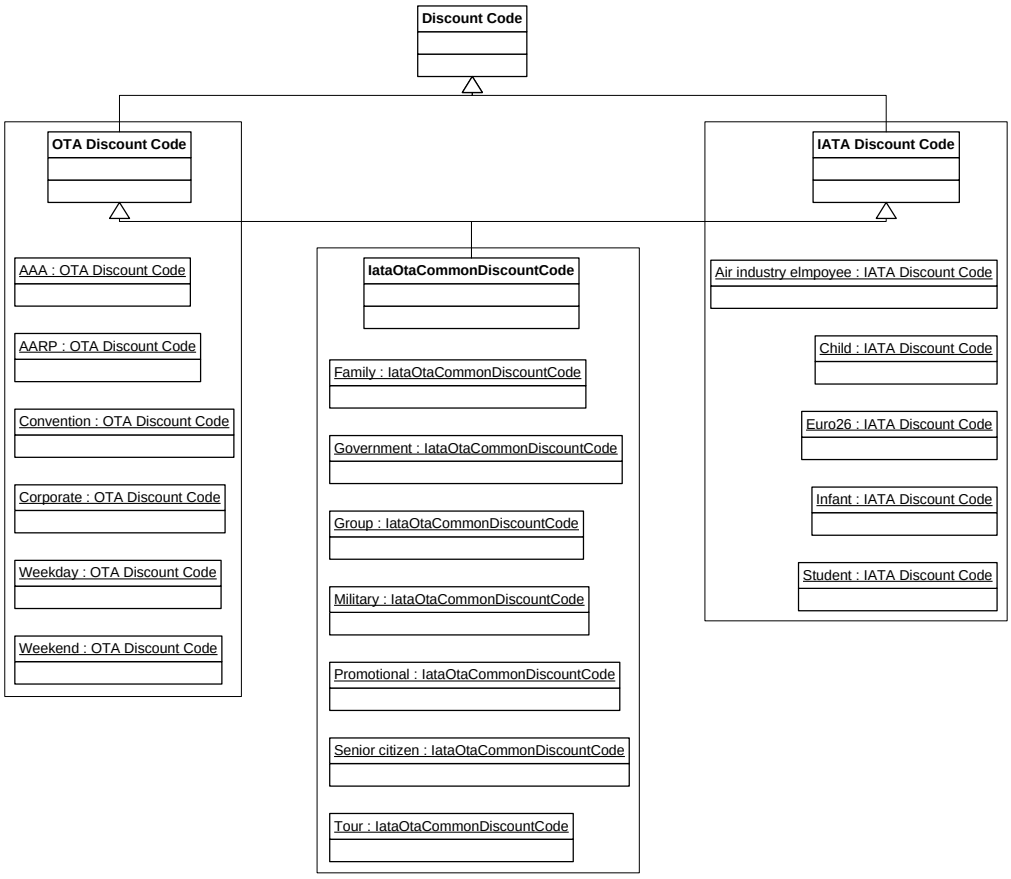

Figure 5. All discount code classes with the distinguished common discount codes.

Presented examples illustrate differences in the integration process for the property and the instance equivalence. In the first example common properties were promoted to become properties of the super-class of classes that shared them – a general class was defined. In the case of discount codes, equivalent instances were demoted to instances of a new class, sub-classes of which represented types of common resources. The latter example illustrates also a method to deal with concepts equivalence definition without explicit statement of this relation; such as the OWL property: *sameAs*.

4. Reuse of external ontologies

The idea of the Semantic Web has been advocated since 1998 [9, 10]. It is based on reuse of existing schemas, which requires less time and resources than designing and implementing new ontologies from scratch. Existing attempts to provide methods and utilities for ontology reusability focus mostly on deciding which concepts are similar and do not investigate technological aspects of the process [36]. Let us therefore illustrate technological issues involved in reusability of external ontologies.

4.1. Reusing Cambia ontology

In the world of travel, currency and payments play an important role, and thus we have considered usage of one of existing currency conversion services:

1. CurrencyExchangeService [3],
2. Currency Conversion Demonstration Web Service [4],
3. CurrencyConvertor [5],
4. Cambia Service [6].

The first three are Web Services (written in .NET) and expose standard WSDL definitions. Additionally, the middle two can be accessed with simple HTTP *post* or *get* request methods. However, utilizing them in the TSS would require development of a completely different communication approach. On the other hand, the latter service exposes FIPA-agent compliant interface, which can be requested through a *FIPA Request Protocol* [27]. Furthermore, the strength of the *Cambia* service lies in the fact it provides ontology for currency conversion. Our original currency ontology allowed only to demarcate the currency code, while the *Cambia* ontology provides detailed conceptualization of the exchange rate, conversion date, currency amount and a hierarchy between these concepts. By utilizing the *Cambia* ontology through a module responsible for communication with *Cambia* service we could:

1. adapt rich currency ontology to our system,
2. extend our system with currency conversion functionality.

Here, we are aiming at:

1. re-using the *Cambia* ontology in our system,
2. specifying currency-related messages to be intelligible to the *Cambia* service.

Cambia ontology was created in Protégé Frames application as an RDF Schema ontology. Interestingly, we have found that it contains also elements from the namespace of Protégé vocabulary, such as: *OverridenProperty* or *min-/max-Cardinality*. This introduces a substantial overhead, as these rules and restrictions are handled **outside of our** TSS. Even more so, this means that *Cambia* ontology is prepared specifically to be used within the context of Protégé created ontologies, which makes it rather inflexible. This dependency can be removed by re-opening the ontology and re-saving it using pure RDFS syntax. This operation can be done either in Protégé Frames tool or programmatically using Protégé Java API. It reduces the size of the

ontology from 345 to 117 statements. Also, as opposed to the original currency ontology, this modified version **is** compatible with the 2.3 version of the Jena framework [30]. More specifically, *Cambia* ontology was implemented with an older version of RDFS (*http://www.w3.org/TR/1999/PR-rdf-schema-19990303*) which resulted in Jena Framework ignoring any RDFS concept utilized in the ontology. When we tried to create a new model object of the original ontology we could not access any class defined in this ontology through the standard interface *OntModel*. Overall, until we have corrected the RDFS namespace, we could not access RDFS resources of *Cambia* currency conversion ontology within Jena 2.3.

Let us also note that *Cambia* ontology was designed in order to easily generate ontology java classes in Protégé with the BeanGenerator plugin [28]. These classes can be then used to create FIPA SL demarcated content [29] of ACL messages which are understandable by the *Cambia* service. Structure of each property in such an ontology defines the *rdfs:range* to be any *rdfs:Class*. The specific range is defined by the Protégé system property *allowedParents*. The latter property is not interpreted natively by Jena. Moreover, when re-saving the original *Cambia* ontology in pure RDFS syntax with the Protégé application, the transformation from:

```
:propertyA
  rdfs:range rdfs:Class;
    protegeSystemNS:allowedParents rdfs:RangeClassOfPropertyA.
```

to:

```
:propertyA
  rdfs:range rdfs:RangeClassOfPropertyA.
```

was not performed. As a consequence, neither the original nor the transformed ontology has clearly defined property ranges according with the RDF(S) specification. Let us summarize this analysis in Table 2.

Feature	Cambia service ontology	Transformed Cambia service ontology
RDFS syntax recognition by Jena 2.3	No	Yes
Size of the overhead	228 statements	0 statements
RDFS syntax recognition by BeanGenerator	Yes	Yes
Syntax standard	RDF(S) + Protégé system vocabulary	RDF(S)
Property ranges visible in Protégé	Yes	No
Property ranges visible in Jena 2.3	No	No

Table 2. Experiment of Cambia ontology reuse summary

4.2. Reusing OWL ontology

As ontology languages are getting more and more mature and number of existing, comprehensive ontologies grows, a need to reuse an existing OWL ontology in a system might appear. There are two main issues that arise in the general case, when employment of OWL ontology into as system like ours is considered: (1) whether to extend our system with the module for operating OWL ontologies or (2) provide semantic level translation between OWL and RDFS.

Our system can access and operate on the ontology through the *OntModel* interface (from the Jena framework) using methods which refer to classes and properties, but when one would like to refer to OWL ontology resources, different methods would have to be used, especially to access OWL-specific properties such as: data-type, object, ontology and annotation properties etc. [8, 9]. However, building or upgrading a special module is time and resource consuming operation. This could also lead to lost of consistency between ontologies written in different languages.

For efficient reuse of the existing OWL ontology we propose its translation to RDFS. Rules for this translation should be simple and limited to translation of all types of classes from OWL to resources defined as *rdfs:Class* and, similarly, different kinds of properties defined in OWL ontology would become resources of *rdf:Property* type.

Simple web application which enables such a mapping can be found at [1]. It has been tested and it produces valid RDFS ontologies which can be read and managed through Jena. Although it is said to be based on Protégé API [2], our survey has shown that Protégé application itself does not support mapping from OWL to RDFS.

Alternatively, transformation which is being discussed can be performed by an adequate definition of *Extensible Stylesheet Language Transformation* (XSL-T).

5. Concluding remarks

In this paper we have addressed practical issues involved in ontology management. In particular, we were interested in merging ontologies describing common aspects of the world (e.g. world of travel); and in re-use of existing ontologies. We have discussed how we merged ontologies of hotel, restaurant and air travel and believe that this approach, while possibly not highly scalable (it is difficult to envision it working smoothly with ontologies consisting of millions of concepts), can be successful. In the case of ontology re-use we have shown that changes in the tools can easily make ontology prepared using one tool completely unusable with another tool.

Currently we are completing the three-ontology-integration process. Our future work includes the following: (1) introduction of ontologies of other travel objects, such as: railroad, cinema or theatre, (2) building a currency conversion mediator agent cooperating with *Cambia* service, (3) testing Protégé API in order to build translator OWL to RDFS. We will report on our progress in subsequent publications.

References

1. OWL2RDFS, http://www.cs.vu.nl/cgi-bin/mcaklein/owl2rdfs
2. Onto-tech, http://www.cs.vu.nl/~se/onto-tech.txt
3. CurrencyExchangeService, http://services.xmethods.net:80/soap
4. Currency Conversion Demonstration Web Service, http://www.webcontinuum.net/webservices/ccydemo.asmx
5. CurrencyConvertor, http//www.webservicex.net/CurrencyConvertor.asmx
6. Cambia Service, http://zurich.agentcities.whitestein.ch/Services/Cambia.html
7. Jena 2 Ontology API – General concepts, http://jena.sourceforge.net/ontology/index.html#generalConcepts
8. Jena Documentation, http://jena.sourceforge.net/documentation.html
9. Semantic Web Road map, http://www.w3.org/DesignIssues/Semantic.html
10. The original proposal of the WWW, HTMLized, http://www.w3.org/History/1989/proposal.html
11. W3C Semantic Web, http://www.w3.org/2001/sw/
12. XML Schema PART 0: Primer Second Edition, http://www.w3.org/TR/xmlschema-0/
13. W3C XML Schema, http://www.w3.org/XML/Schema
14. XML Schema Tutorial, http://www.w3schools.com/schema/default.asp
15. OWL Web Ontology Language – Use Cases and Requirements, http://www.w3.org/TR/webont-req/
16. OWL Web Ontology Language – Overview, http://www.w3.org/TR/owl-features/
17. Introduction to OWL, http://www.w3schools.com/rdf/rdf_owl.asp
18. Davis J., Fensel D., Harmelen F.: Towards the Semantic Web: Ontology-Driven Knowledge Management, John Wiley & Sons, (2003).
19. Fensel D.: Ontologies: A Silver Bullet for Knowledge Management and Electronic Commerce, Springer, Berlin, (2001).
20. Gawinecki M., Gordon M., Nguyen N. T., Paprzycki M., Szymczak M.: RDF Demarcated Resources in an Agent Based Travel Support System. In: M. Golinski et. al. (eds.), Informatics and Effectiveness of Systems, PTI Press, Katowice, (2005), 303-310.
21. Gawinecki M., Gordon M., Nguyen N. T., Paprzycki M., Vetulani Z.: Ontologically Demarcated Resources in an Agent Based Travel Support System. In: R. K. Katarzyniak (ed.) Ontologies and Soft Methods in Knowledge Management, Advanced Knowledge International, Adelaide, Australia, (2005), 219-240.
22. Vukmirovic M., Szymczak M., Ganzha M., Paprzycki M.: Utilizing Ontologies in an Agent-based Airline Ticket Auctioning System. In: Proceedings of the 28th ITI Conference, (to appear).
23. Gordon M., Kowalski A., Paprzycki A., Pełech T., Szymczak M., Wąsowicz T.: Ontologies in a Travel Support System. In: D. J. Bem et. al. (eds.) Internet 2005, Technical University of Wroclaw Press, (2005), 285-300.
24. Vukmirovic M., Ganzha M., Paprzycki M.: Developing a Model Agent-based Airline Ticket Auctioning System. In: Proceedings fo the IIPWM Conference (to appear).
25. Greenberg J., Sutton S., Campbell D.G.: Metadata: A Fundamental Component of the Semantic Web. In: Bulletin of the American Society for Information Science and Technology, 4(29), (2005), 16-18.

26. XML Information Set (Second Edition), http://www.w3.org/TR/xml-infoset/
27. FIPA Request Interaction Protocol Specification, http://www.fipa.org/specs/fipa00026/PC00026D.pdf
28. Beangenerator, http://acklin.nl/beangenerator/
29. FIPA SL Content Language Specification, http://www.fipa.org/specs/fipa00008/SC00008I.html
30. Jena Semantic Web Framework, http://jena.sourceforge.net/
31. International Air Transport Association, http://www.iata.org/
32. IATA Airline Coding Directory – Airline Destignators, Dec 1, 2005 until Dec 1, 2006
33. IATA Standard Schedules Information Manual, Mar 1, 2006 until Sep 30, 2006
34. OpenTravel Alliance, http://www.opentravel.org/
35. Baclawski K.: Versatile Information Systems - Tutorial on the Semantic Web
36. Fernández M., Cantador i., Castells P.: CORE: A Tool for Collaborative Ontology Reuse and Evaluation. In: 4th International Workshop on Evaluation of Ontologies for the Web (EON 2006) at the 15th International World Wide Web Conference (WWW 2006). Edinburgh, UK, (2006).
37. IATA Reservations Service Manual, 23rd Edition, Effective 1st June, 2006.
38. Vukmirovic M., Szymczak M., Ganzha M., Paprzycki M.: Utilizing Ontologies in an Agent-based Airline Ticket Auctioning System. In: Proceedings of the 28th ITI Conference (to appear)
39. Vukmirovic M., Ganzha M., Paprzycki M.: Developing a Model Agent-based Airline Ticket Auctioning System. In: Proceedings of the IIPWM Conference (to appear)

Maciej Gawinecki
Systems Research Institute, Polish Academy of Science
Newelska 6, 01-447 Warsaw, Poland
e-mail: maciej.gawinecki@ibspan.waw.pl

Michał Szymczak
Department of Mathematics and Computer Science, Adam Mickiewicz University
Umultowska 87, 61-614 Poznań, Poland
e-mail: d124124@wmid.amu.edu.pl

Mladenka Vukmirovic
Industry Development Department, Montenegro Airlines
Beogradska 10, 81000 Podgorica, Montenegro
e-mail: mladenka.vukmirovic@mgx.cg.yu

Marcin Paprzycki
Computer Science, SWPS
Chodakowska 18/31, 03-815 Warszawa, Poland
e-mail: marcin.paprzycki@swps.edu.pl