Knowledge Integration for Domain Modeling[1]

Armands Slihte, Janis Osis and Uldis Donins

Faculty of Computer Science and Information Technology
Institute of Applied Computer Systems, Riga Technical University, Riga, Latvia

Abstract. This research integrates artificial intelligence (AI) and system analysis by exploiting ontology, natural language processing (NLP), business use cases and model-driven architecture (MDA) for knowledge engineering and domain modeling. We describe an approach for compounding declarative and procedural knowledge in a way that corresponds to AI and system analysis standards, and is compliant for acquiring a domain model corresponding to MDA standards. We are recognizing the possibility of automatically transforming this knowledge to a Computation Independent Model (CIM) for MDA.

1 Introduction

Computer science has come a far way in understanding knowledge and developing means to manage knowledge. Knowledge engineering is mostly associated with artificial intelligence (AI), but many aspects of system analysis also deal with it. There have been significant results and applications in both artificial intelligence (AI) and system analysis. On the other hand, the integration between these two domains and the benefits it can offer has not yet been fully recognized. There are few approaches that have been going in this direction; we analyze these in the related work section. Nevertheless, none of these approaches suggest a solution for acquiring the domain model automatically from the corresponding domain knowledge, which should be the case. There is no reason why we could not automatically generate a model for a domain, for which we have all the corresponding knowledge explicitly defined.

The approach proposed in this paper provides a formal way to facilitate ontology for software engineering, more specifically for acquiring a Computation Independent Model (CIM) within Model Driven Architecture (MDA). It does not suggest a novel methodology for ontology development, but instead is based on the existing methodologies. This approach suggests ontology to be directly used as an input for domain modeling by exploiting business use cases and natural language processing (NLP). We are using Web Ontology Language (OWL) and Protégé 4.1 tool for ontology development, and Attempto Controlled English (ACE) for natural language processing. We are using Topological Functioning Model (TFM) as the CIM within MDA; acquiring a mathematically formal and thus transformable CIM. This is dis-

[1] This work has been supported by the European Social Fund within the project. Support for the implementation of doctoral studies at Riga Technical University".

Slihte A., Osis J. and Donins U..
Knowledge Integration for Domain Modeling .
DOI: 10.5220/0003581200460056
In *Proceedings of the 3rd International Workshop on Model-Driven Architecture and Modeling-Driven Software Development* (MDA & MDSD-2011), pages 46-56
ISBN: 978-989-8425-59-1

cussed in more detail in section 5 – Integration with MDA. In this paper we also describe the approach using an example of a library business system, to show how this approach can be applied. Moreover, a long-term goal of this research is to provide a corresponding toolset to support this approach, so that the automation of domain modeling could be complete.

This paper is organized as follows. Section 2 considers related work for cooperating knowledge engineering and system analysis. Section 3 distinguishes between declarative and procedural knowledge and suggesting. Section 4 analyzes the knowledge representation possibilities and focuses on controlled natural language, ontology and business use cases. Section 5 explains the approach for integrating knowledge for domain modeling and provides an example of a library business system. Section 6 refers to a methodology for integrating the suggested approach with MDA.

2 Related Work

This work continues research on computation-independent modeling and specifically on TFM for MDA started in [1], [2], [3] and [4]. As stated in [4] an informal description of the system in textual form can be produced as a result of system analysis. This approach proposes to transform a system's informal description into a TFM of the system. In this paper we show how to go even further and use formally defined knowledge as input for generating TFM.

Other authors have been investigating how to combine AI and system analysis for the benefit of domain modeling, incorporating ontologies with MDA. Ontologies, as formal representations of domain knowledge, enable knowledge sharing between different knowledge-based applications. Diverse techniques originating from the field of artificial intelligence are aimed at facilitating ontology development. However, these techniques, although well known to AI experts, are typically unknown to a large population of software engineers [5].

In order to overcome the gap between the knowledge of software engineering practitioners and AI techniques, a few proposals have been made suggesting the use of well-known software engineering techniques, such as UML, for ontology development. An approach proposed in [6] is dealing with generating Resource Description Framework (RDF) from a UML model. RDF is a W3C XML-based standard for sharing ontologies on the Semantic Web. Another approach [7] proposes a transformation to semantic extraction of ontologies from UML models. Their initial presumption is that UML and ontologies complement each other. That is to say, UML is designed for building models by human experts, while OWL is designed to be used at run time by intelligent processing methods.

The 2 approaches mentioned earlier are useful if you already have the design model (UML) and want to acquire the ontology. From the perspective of MDA the order of the acquired artifacts is incorrect, because the design model or PIM/PSM should be derived from CIM, which includes the declarative knowledge provided by an ontology. This means that the ontology has to come first, in order to construct an accurate PIM/PSM. In this paper we insist on starting with knowledge and not the design.

An independent ontology metamodel using the MOF has been developed in [8]; it is named the Unified Ontology Language (UOL). This is important from the perspective of MDA. OWL is a well-known standard for ontology development, but the transformation between OWL and a model defined according to MOF would not correspond to MDA standards. On the other hand, this newly introduced UOL still needs an ad-hoc transformation mechanism from OWL. Nevertheless, UOL could be considered in further stages of this research as a format for ontology.

3 Declarative and Procedural Knowledge

The traditional Artificial Intelligence (AI) techniques most frequently used to represent knowledge in practical intelligent systems include object-attribute-value triplets, uncertain facts, fuzzy facts, rules, semantic networks, and frames. Ontologies have acquired major importance in knowledge representation as well [5]. On the other hand, system analysis researchers have developed means to manage knowledge about business systems and processes, e.g. Business Process Modeling Notation (BPMN) and Model Driven Architecture (MDA).

Knowledge means understanding of a subject area. It includes concepts and facts about that subject area, as well as relations among them and mechanisms for how to combine them to solve problems in that area [5]. The term knowledge can be used to refer to a state of knowing facts, methods, principles, techniques and so on. This common usage corresponds to what is often referred to as “know about”. Second, usage of the term knowledge is when it refers to understanding facts, methods, principles and techniques sufficient to apply them in the course of making things happen. This corresponds to “know how”. Cognitive psychologists sort knowledge into two categories: declarative and procedural [9]. From the perspective of a student who is learning knowledge: 1) Declarative knowledge is that the student knows or understands (e.g. Riga is the capital of Latvia, book catalogue has entries of books); 2) Procedural knowledge is that the student is able to do something (e.g. buy an airplane ticket to Riga, get a book at the library).

This distinction between declarative and procedural knowledge may seem obvious, but has not been recognized too often. To properly describe a business system in its environment, it is necessary to know both – declarative and procedural knowledge. Approaches like BPMN and MDA are very strong describing the procedural knowledge, but lack the AI strength in dealing with declarative knowledge. In this paper we propose an approach, which integrates declarative and procedural knowledge providing a common approach for system analysis with the perspective of integrating with MDA.

4 Knowledge Representation

Before any system analysis process can start, it is necessary to acquire knowledge about the business system and its environment. Most of this knowledge usually is defined in different documents in a form of natural language. It is necessary to store this knowledge in a way, so that it could be understandable by a computer.

Attempto Controlled English (ACE) is a controlled natural language, in other words it is a subset of English with a restricted syntax and a restricted semantics described by a small set of construction and interpretation rules. It is a formal language and can automatically and unambiguously be translated into first-order logic. Although ACE may appear perfectly natural it can be read and understood by human and machine. One could say that ACE is a first-order logic language with the syntax of a subset of English. ACE can be used as knowledge representation, specification and query language [10]. ACE was originally intended to specify software, but has since been used as a general knowledge representation language in several application domains. With Attempto Parsing Engine (APE) it is possible to derive a syntax tree from ACE texts which is crucial for Topological Functioning Model (TFM) approach.

To someone who wants to discuss topics in a domain D using a language L, ontology provides a catalogue of the types of things assumed to exist in D; the types in the ontology are represented in terms of the concepts, relations, and predicates of L [11]. Some reasons for developing an ontology are: 1) To share common understanding of the structure of information among people or software agents; 2) To enable reuse of domain knowledge; 3) To make domain assumptions explicit; 4) To separate domain knowledge from operational knowledge; 5) To analyze domain knowledge [12]. Ontologies are used for different purposes, but this research focuses on ontologies developed for a business domain, describing business terms and their relationships.

Ontology is a perfect candidate for representing declarative knowledge about a business system and its environment. Ontology defines the terms used to describe and represent an area of knowledge. Ontologies include computer-usable definitions of basic concepts in the domain and the relationships among them. OWL [13] is a common standard for defining ontologies and will be considered for further knowledge integration for domain modeling.

Business use cases are not normalized or standardized by any consortium, unlike UML use case diagram, which is defined by Object Management Group. Business use cases should not be mistaken with UML use case diagram. Moreover, there are many different use case templates and the structure of a use case can be adjusted depending on the situation and the development team [14]. These textual business use cases are considered for representing the procedural knowledge. The following structure of a use case is considered: 1) use case title, 2) actors, 3) pre-conditions, 4) main scenario, 5) extensions, and 6) sub-variations. Using ACE is considered for defining the step of the business use cases [10]. Business use cases provide a formal data structure that can be used to represent the procedural knowledge about a business system. By using ACE we enable this knowledge to be processed by a computer.

5 Integrating Knowledge for Domain Modeling

The steps of the Business Use Cases are defined using the ACE. This solves some of the natural language problems, but not all. Still ACE texts do not solve possible ambiguity. ACE doesn't restrict the usage of nouns and it is possible to express the same meaning using different words. Another possible problem is the inconsistency of business use cases. There might be steps defined that do not make sense in the given business system or its environment. If there would be a predefined lexicon for the specific domain, it would be possible to deal with these problems. Ontology can be used as this lexicon. In this section we are proposing an approach for integrating declarative and procedural knowledge. By exploiting ontology we prevent the ambiguity and inconsistency of business use cases.

5.1 Exploiting Ontology

The ontology is an extremely important part of the knowledge about any domain. Moreover, the ontology is the fundamental part of the knowledge, and all other knowledge should rely on it and refer to it.

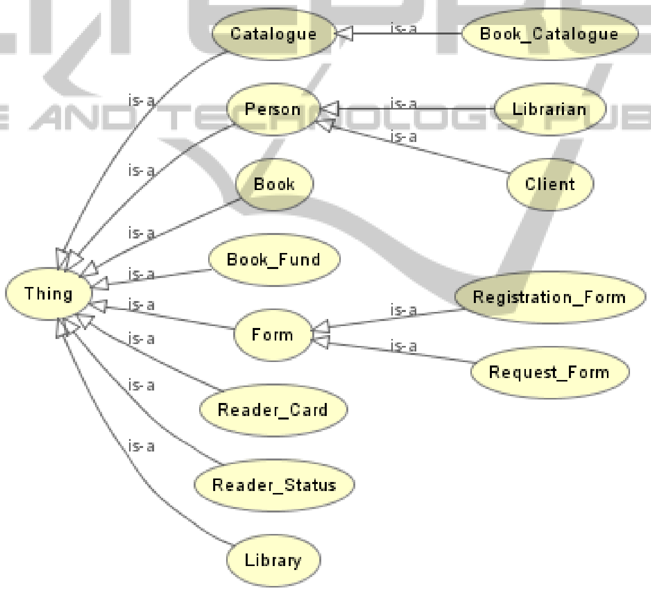
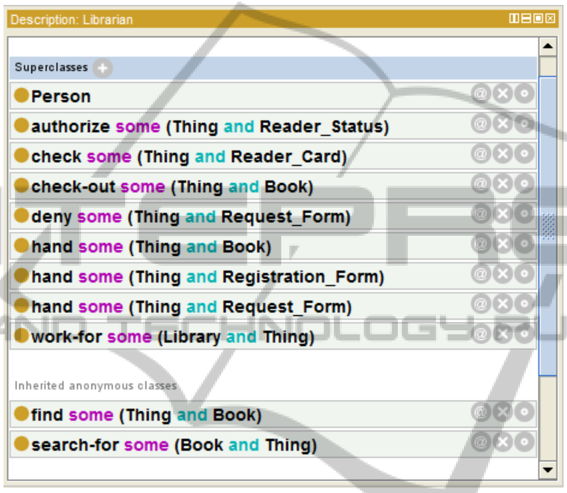


Fig. 1. This is the class hierarchy of an ontology for a library. It includes the main classes that a library business system needs to function. The main actors are the Client and the Librarian, which both are sub-classes of Person. Class hierarchy includes also important concepts for a Library business system like Library, Book, Book Catalogue, Reader Card and Request Form.

This work does not propose a methodology for developing ontologies, but a methodology to use an already developed ontology for further knowledge engineering and system analysis. If there is no ontology defined for the business system before, it is necessary to build by analyzing the business system and its environment. Available documents and expert knowledge is the main input for this development. Particularly in the case of information extraction almost every text introduces new terms, so we cannot assume that all terms encountered in the text we process will already be included in the ontology. The ability to add new terms to an existing ontology is crucial even when using an ontology whose structure has been formally defined [15]. This means that even if we initially have a defined ontology, it might lack some classes and properties for the business system or its environments under consideration.

To show how our approach will be using ontology for integrating declarative and procedural knowledge and after that using this knowledge for domain modeling, authors of this paper consider a library business system as example. In the following figures you can see the ontology considered.



Description: Librarian

Superclasses

Person
authorize some (Thing and Reader_Status)
check some (Thing and Reader_Card)
check-out some (Thing and Book)
deny some (Thing and Request_Form)
hand some (Thing and Book)
hand some (Thing and Registration_Form)
hand some (Thing and Request_Form)
work-for some (Library and Thing)

Inherited anonymous classes

find some (Thing and Book)
search-for some (Book and Thing)

Fig. 2. This is the description of the librarian class. You can see some of the properties and relationships between properties and classes. For example, one of the properties is checking out a book from a book fund, which is done by the librarian.

Developing an ontology includes: 1) defining classes in the ontology; 2) arranging the classes in a taxonomic hierarchy; 3) defining properties and describing the relationships with classes; 4) defining the individuals. Creating the class hierarchy is the first and second step (Fig. 1 shows an example). This ontology for an abstract library business system was developed by the authors of this paper to show an example. The third step is defining the properties and relationships between classes and properties (Fig. 2 shows an example). It is important that all the business system's concepts and actions that can be associated with these concepts are defined. Recognizing a satisfactory scope for the domain is not easy. It will not always be possible to capture everything on the first try, but as mentioned before this has to be an iterative process. The ability to modify the ontology is crucial, because the scope of the domain can also change.

5.2 Developing Business Use Cases

Ontologies provide logical statements that describe what terms are, how they are related to each other, and how they can or cannot be related to each other. Business use cases on the other hand provide a formal way to define the procedural knowledge, showing step by step how a process is executed, what the variations are and which actors are involved. The problem with business use cases is that their steps are sentences in natural language. We are restricting them by applying ACE, which guarantees we can analyze this sentence by syntax and get a parse tree.

Use case: **Requesting a book**
Actors: Client, Librarian
Preconditions: Librarian authorizes a reader status
Main scenario:
1. Client searches for a book in a catalogue
2. Librarian hands a request form
3. Client fills a request form
4. Librarian checks out a book from a book fund
5. Librarian hands a book to a client
6. Client leaves a library

Extensions:
1a. If client hasn't found a book, client leaves a library

Sub-variations:
4a. If a book is checked out, librarian denies a book request form
4a1. Client searches for a book in a catalogue
6a. If client wants another book, client searches for a book in a catalogue

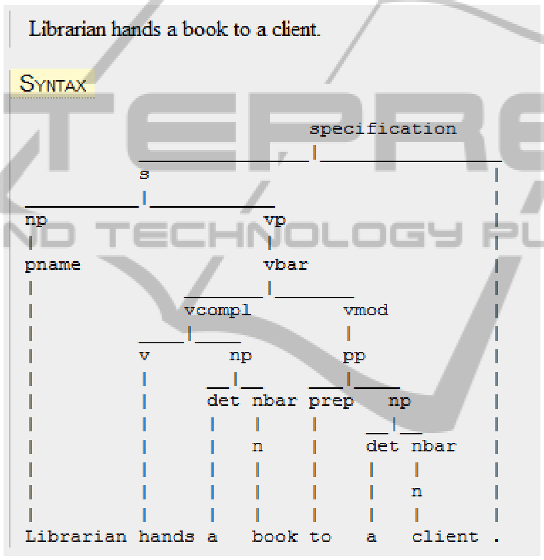
Fig. 3. This is a business use case for requesting a book in a library business system. First a client searches for a book in the catalogue and then fills a request form to get the book. Librarians responsibility is to hand the request form, check out the book from the book fund and hand the book to the client. There are also variations.

Nevertheless, it does not guarantee that the terms used in sentences will be unambiguous. For example, there could be steps "Client fills a request form" and "Librarian denies a form". These steps are correct from a syntax perspective, but they are inconsistent, because in the first sentence a form that is meant for requesting a book is defined as "Request form", but in the second sentence it is defined as "Form". This would not be a problem if there was a predefined vocabulary, which determines that these terms mean the same thing in this domain.

Controlled natural language also does not guarantee that the step will make sense for the given domain. For example, there could be a step "Librarian shows a reader card". This step is perfectly correct from a perspective of syntax and may seem to make sense, because librarian can also be a reader in a library, but for the given domain "Librarian" is a definition of the person who works for the library and at this moment in time is fulfilling this role. So actually this sentence does not make sense from the perspective of the domain.

We cannot put this much responsibility on a system analyst who will be developing these business use cases. This kind of ambiguity and inconsistency should be automatically discovered and eliminated. The approach suggested in this paper will use ontology to solve both problems – the possible ambiguity and inconsistency of the sentences. Please consider the business use case shown in Fig. 3.

For the first problem of ambiguity, let us look at the first step "Client searches for a book in a catalogue". We could rephrase this also like this "Client searches for a book in a book catalogue". Notice that in the second sentence we specify that it is a book catalogue and not just any catalogue. In this specific domain both sentences refer to the same object and it is important, that when the steps get analyzed the correct objects are considered. If we look at the library ontology's class hierarchy (Fig. 1), it is clearly defined that "Catalogue" is a super-class of "Book catalogue". This solves the problem of ambiguity in this case, because we know it refers to the same object. Another problem appears if we rephrase the step like this "Client searches for literature in a catalogue". The concept "Literature" is not defined in our ontology, so this sentence should be marked as invalid until someone defines the concept in the ontology. The same applies to the properties. If the sentence is "Client looks for a book in a catalogue" and property "look" is not defined, this step should be marked as invalid.



Librarian hands a book to a client.

SYNTAX

```
                         specification
          _______________|________________
          s                              |
__________|___________                   |
np                   vp                  |
|                    |                   |
pname                vbar                |
|             _______|_______            |
|             vcompl        vmod         |
|         ____|____         |            |
|         v       np        pp           |
|         |     __|__    ___|____        |
|         |     det nbar prep   np       |
|         |     |   |    |    __|__      |
|         |     |   n    |    det nbar   |
|         |     |   |    |    |   |      |
|         |     |   |    |    |   n      |
|         |     |   |    |    |   |      |
Librarian hands a   book to   a   client .
```

Fig. 4. This is a parse tree generated by ACE parser from a business use case step sentence. In the syntax s – sentence; np – noun phrase; cp – verb phrase; pname – proper name; vcompl – verb with complement; vmod – verb phrase modifiers; v – verb; pp – prepositional phrase; det – determiner; n –noun.

For the second problem of inconsistency let us consider the example mentioned earlier "Librarian shows a reader card". From our ontology's property and class relationships (Fig. 2) we see that the relationship between "Librarian" and "Reader card" is defined by property "check" and not "show". This implies that the sentence is invalid and should be corrected or the ontology has to be modified. By checking the correspondence between properties and classes it is possible to deal with this problem.

To implement the solution for these problems technically we will be analyzing the parse trees of the sentences. Fig. 4 shows a parse tree the business use case step "Librarian hands a book to a client". This sentence can be broke down into verb phrase, noun phrase and prepositional phrase, and then also into verbs and nouns. Approach for knowledge integration suggests that the nouns need to correspond to the classes and the verbs need to correspond to the properties. If they do not, then an error should be raised and either the ontology or the business use case needs to be modified. For this example we see that “Librarian”, ”Book”, “Client” and “hand” is defined by the library ontology, so we have a valid sentence from perspective of ambiguity – all terms are defined and understandable. From perspective of consistency “Librarian” and “Book” are associated with “hand”. This association can be confirmed by the ontology, because a librarian hands a book.

6 Integration with Model Driven Architecture

In previous work [1], [2], [3] and [4] authors introduce an algorithm to automatically derive the TFM from textual use cases of a business system. Same business use cases structure is used to define the procedural knowledge. This algorithm utilizes the statistical parser to analyze the syntax of use case sentences and identify functional features for the TFM. The problem there is the potential ambiguity and inconsistency of the business use case steps, which authors are solving by applying ontology in this paper.

TFM offers a formal way to define a system by describing both the system’s functional and topological features. TFM is represented in the form of a topological space (X, Θ), where X is finite set of functional features of the system under consideration, and Θ is the topology that satisfies axioms of topological structures and is represented in the form of a directed graph [4]. TFM represents the system in its business environment and shows how the system is functioning, without details about how the system is constructed. This research considers TFM to be CIM within MDA; acquiring a mathematically formal and thus transformable CIM.

The integration with MDA is already defined with the algorithm for deriving TFM from business use cases [3]. Other branch of this research is suggesting a TopUML profile, which incorporates the topological nature of TFM with UML. This provides unique benefits for MDA, because it is possible to acquire cause-effect relationships between methods for PIM/PSM from CIM [16].

7 Conclusions

This paper describes a novel approach for integrating AI and system analysis by facilitating ontology, natural language processing, business use cases and MDA. This approach provides a way for acquiring the domain model automatically from the corresponding domain knowledge. It provides a formal way to use ontology for system analysis and suggests ontology to be directly used as an input for domain modeling by exploiting business use cases and natural language processing. This knowledge can be used for generating CIM according to previous research on TFM.

Future research includes: 1) developing guidelines for identifying the scope of the domain; 2) developing an algorithm for business use case step ambiguity and inconsistency checking according to given ontology; 3) developing guidelines for ontology development or defining the supported ontology development methodologies; 4) implementing a tool for business use cases development, which would take OWL as input; 5) Integrating this business use case development tool with TFM generation and TopUML tools.

This approach provides a new perspective for domain modeling, allowing the domain model to be generated from formally defined knowledge; thus exploiting the power of knowledge engineering, leaving less space for interpretation and enhancing MDA with a formal CIM.

References

1. Šlihte A.: The Specific Text Analysis Tasks at the Beginning of MDA Life Cycle. In: Databases and Information Systems Doctoral Consortium, Latvia, Riga, 5.-7. July (2010) 11–22.
2. Šlihte A.: Implementing a Topological Functioning Model Tool. In: Scientific Journal of Riga Technical University, 5. series., Computer Science, Vol. 43, Riga (2010) 68–75
3. Šlihte A.: Transforming Textual Use Cases to a Computation Independent Model. MDA & MTDD 2010, Greece, Athens, 22.-24. July (2010) 33–42.
4. Osis, J., Asnina, E., Grave, A.: Computation Independent Representation of the Problem Domain in MDA. J. Software Eng. Vol. 2, Iss. 1, (2008) 19—46 Available: http://www.e-informatyka.pl/e-Informatica/Wiki.jsp?page=Volume2Issue1 [Accessed: Feb 28, 2010].
5. Gasevic, D., Djuric, D., Devedzic, V.: Model Driven Architecture and Ontology Development. Springer, Heidelberg (2006).
6. Cranefield, S.: Networked knowledge representation and exchange using UML and RDF. Journal of Digital Information, Vol. 1, No. 8 (2001) Available: https://journals.tdl.org/jodi/article/viewArticle/30/31 [Accessed: Feb 28, 2011].
7. Falkovych, K., Sabou, M., Stuckenschmidt, H.: UML for the Semantic Web: Transformation-Based Approaches. In B. Omelayenko and M. Klein, editors, Knowledge Transformation for the Semantic Web. IOS Press (2003) 92–106 http://www.cwi.nl/˜media/publications/UML_for_SW.pdf [Accessed: Feb 28, 2011].
8. Baclawski, K., Kokar, M. K., Kogut, P., Hart, L., Smith J. E., Letkowski, J., Emery, P.: Extending the Unified Modeling Language for Ontology Development. Int. Journal Software and Systems Modeling (SoSyM) Vol. 1, No. 2 (2002) 142-156.
9. Poesio, M.: Domain modelling and NLP: Formal ontologies? Lexica? Or a bit of both? Applied Ontology, Vol. 1, No. 1. IOS Press (2005) 27–33.
10. Noy, N. F., McGuinness, D. L.: Ontology development 101: A guide to creating your first ontology. Technical Report SMI-2001-0880, Stanford Medical Informatics (2001).
11. Nickols, F.: The knowledge in knowledge management (KM). in J.W. Cortada and J.A. Woods, The Knowledge Management Yearbook 2001–2002, Butterworth-Heinemann, Boston (2000).
12. Fuchs, N. E., Kaljurand, K., Kuhn, T.: Attempto Controlled English for Knowledge Representation. In Cristina Baroglio, Piero A. Bonatti, Jan Maluszynski, Massimo Marchiori, Axel Polleres, and Sebastian Schaffert, editors, Reasoning Web, Fourth International Summer School 2008, Lecture Notes in Computer Science 5224. Springer (2008) 104–124.
13. W3C, OWL Web Ontology Language Overview, W3C Recommendation February 10 2004. Available: http://www.w3.org/TR/owl-features/ [Accessed: Feb 28, 2011].
14. Malan, R., Bredemeyer, D.: Functional Requirements and Use Cases, March 2001. Available: http://www.bredemeyer.com/pdf_files/functreq.pdf [Accessed: Feb 28, 2011].

15. Guarino, N.: Formal Ontology and Information Systems. Proceedings of Formal Ontology and Information Systems, Trento, Italy, IOS Press, Amsterdam (1998) 3–15.
16. Donins, U.: Software Development with the Emphasis on Topology. In: Proceeding of 13th East-European Conference on Advances in Databases and Information Systems (ADBIS 2009). Volume 5968 of LNCS. Springer (2010) 220-228.