Ontologies in Checking for Inconsistency of Requirements Specification

Kroha, P., Janetzko, R.
TU Chemnitz, Faculty of Computer Science, Chemnitz, Germany
Email: kroha@informatik.tu-chemnitz.de, robert_janetzko@web.de

Labra, J.E.
E.U. de Ingenieria Tec. Informatica de Oviedo, Universidad de Oviedo, Oviedo, Spain
Email: jelabra@gmail.com

Abstract

In this paper, we investigate how the methods developed for using in Semantic Web technology could be used in validating requirements specifications.

The goal of our investigation is to do some (at least partial) checking and validation of the UML model using a predefined domain-specific ontology in OWL, and to process some checking by using the assertions in descriptive logic.

We argue that the feedback caused by the UML model checked by ontologies and OWL DL reasoning has an important impact on the quality of the outgoing requirements.

This paper describes not only methods but also the implementation of our tool TESSI (in Protégé, Pellet, Jess) and practical experiments in consistency checking of requirements.

1. Introduction

Requirements specification is a complex and time-consuming process. The goal is to describe exactly what the user wants and needs. Any failure and mistake in requirements specification is very expensive because it causes the development of software parts that are not compatible with the real needs of the user and must be reworked later for additional costs.

When the analysis phase of a project starts, analysts have to discuss the problem to be solved with the customer (users, domain experts) and then write the requirements found in form of a textual description. This is a form the customer can understand. However, any textual description of requirements can be (and usually is) incorrect, incomplete, ambiguous, and inconsistent. Later on, the analyst specifies a UML model based on the requirements description he has written himself before. However, users and domain experts cannot validate the UML model as most of them do not understand (semi-)formal languages such as UML. Misunderstanding between analysts and users is very common and brings projects over budget.

Requirements are based on knowledge of domain experts and users' needs and wishes. One possible way to classify this knowledge and then fashion it into a tool is through ontology engineering. Simplified, ontologies are structured vocabularies(basic concepts in the domain and relations among them) having additionally the possibility of reasoning.

This approach will be discussed in this paper. For our requirements specification tool TESSI, we have designed and implemented an ontology-based component that helps to reduce misunderstanding, missed information, and helps to overcome some of the barriers that make successful specification of requirements so difficult. In this paper we focus on checking consistency of requirements.

The rest of the paper is organized as follows. In Section 2 we discuss related work. In Section 3 we briefly explain why ontology is the proper mechanism to be used in requirements specification. Section 4 describes the architecture and functionality of our implemented requirements specification tool TESSI. In Section 5 the implementation is given. In Section 6 we present the results of experiments, Section 7 contains conclusions.

2. Related work

Using ontologies to shape the requirements engineering process is clearly not a new idea. In the area of knowledge engineering, ontology was first defined by [25].

An ontology-based approach to knowledge acquisition from text through the use of natural language recognition is discussed in [2], [17], in [19] and the last approach in [4]. In [30] they have constructed the Enterprise Ontology to aid developers in taking an enterprise-wide view of an organisation. The approach in [17] is intended to automate both interactions with users and the development of application models.

The ontologies used by [28] in their Oz system are domain models which prescribe detailed hierarchies of domain objects and relationships between them. Formal models for ontology in requirements are described in [16]. Domain rules checking is described in [24]. In [32] the inconsistency measurement is discussed. The ontology used by the QARCC system [1] is a decomposition taxonomy of

software system quality attributes. In [24] a formal model of requirements elecitation is discused that contains domain ontology checking.

Concerning inconsistencies the overview is given in [5], in [26], and lately in [29] but there is not an approach applying ontology in the sense of our way.

However, our work is not specifically addressing the issue of improving natural language communication between stakeholders in an interview in order to achieve more polished requirements as most of the related papers are. We investigate the possibility of combining UML model and OWL ontology for checking and validating requirements specifications, as we have already mentioned above.

3. Why to Use Ontology for Checking Requirements Specifications

Ontologies seem to be the right tool because they are designed to capture natural language descriptions of domains of interest. An ontology consists of:

- Description part - a set of concepts (e.g. entities, attributes, processes), their definitions and their interrelationships. This is referred to as a conceptualization. Here, ontology represents the domain knowledge (domain ontology) and requirements can be seen as a specialized subset of it (as problem ontology in our text).
- Reasoning part - a logical theory that constrains the intended models of logical language containing:
  - integrity rules of the domain model representing the domain knowledge,
  - derivation rules and constraint rules of the problem model.

  Reasoning in ontologies brings the inferential capabilities that are not present in taxonomies used for modeling formerly. It makes possible to search for contradictions that indicate inconsistencies.

4. The tool TESSI - Architecture and Dataflow

We argue that there is a gap between the requirements definition in a natural language and the requirements specification in some semi-formal graphical representation. The analyst's and the user's understanding of the problem are usually more or less different when the project starts. The first possible point of time when the user can validate the analyst's understanding of the problem is when a prototype starts to be used and tested.

In our approach [20], [21], [23] that will be developed further in this report, we offer a textual refinement of the requirements definition which can be called requirements description. Working with it, the analyst is forced by our supporting tool to complete and explain requirements and to

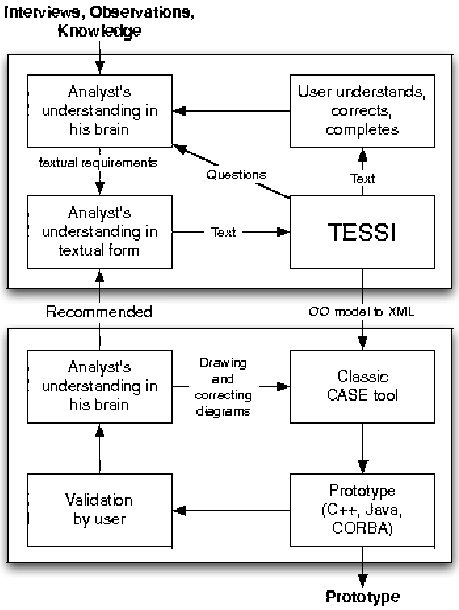

Figure 1. Architecture and dataflow of our tool TESSI

specify the roles of words in the text in the sense of object-oriented analysis. During this process, a UML model will be built with our tool driven by the analyst's decisions. Based on this UML model a new, model-derived requirements textual description will automatically be generated [23] that describes the analyst's understanding of the problem. Now, the user has a good chance to read it, understand it and validate it. His/her clarifying comments will be used by the analyst for a new version of the requirements description. The process repeats until there is a consensus between the analyst and the user. This does not mean that the requirements description is perfect, but some mistakes and misunderstandings are removed.

We argue that the textual requirements description and its preprocessing by our tool will positively impact the quality and the costs of the developed software systems because it inserts additional feedbacks into the development process. In this paper, we present how the UML model is used for checking consistency. There are some other checking possibilities but their presentation is out of the scope of this paper.

Using experiences given in [24], we describe the domain ontology in Protégé and apply ontology reasoning (e.g. the inference engine in Pellet - for checking classes) first for domain ontology checking, then for requirements problem ontology checking, and last for checking whether the requirements problem ontology subsumes the domain ontology.

The steps of the requirements processing that uses ontologies are the following:

- building a domain ontology in OWL using Protégé, i.e.

domain ontology description based on OWL and SWRL based is constructed by domain experts at first and then transformed into the concepts set of Pellet and roles set of Jess.
- checking the domain ontology for consistency using Pellet (class hierarchy) and Jess (rules),
- the analyst writes a text description of requirements based on interviews with users,
- the analyst builds the UML model from a textual description of requirements supported by our tool TESSI,
- conversion of requirements described as a UML model to a problem ontology in OWL using convertor ATL (we used ATL because RacerPro is not public),
- checking the problem ontology for its consistency,
- checking the problem ontology for consistency with the domain ontology,
- identifying inconsistency problems,
- finding the corresponding parts in the former textual description of requirements and correcting them,
- building a new UML model based on corrected textual description of requirements,
- after iterations when no ontology conflicts have been found a new textual description of requirements will be automatically generated that corresponds to the last iteration of the UML model,
- before the UML model will be used for design and implementation the user and the analyst will read the generated textual description of requirements and look for missing features or misunderstandings,
- problems found can start the next iteration from the very beginning,
- after no problems have been found the UML model in form of a XMI-file will be sent to Rational Modeler for further processing.

5. Implementation

As we already mentioned above we needed to implement:
- converting UML model into problem ontology model,
- checking ontology class hierarchy,
- checking consistency of ontology rules.

The component of TESSI containing the ontology-based consistency checking of requirements specification has been implemented in [14].

5.1. Using ATL for Converting UML to OWL

Our goal was to convert the UML model obtained from the textual requirements into a corresponding problem ontology model that can be compared with the domain ontology model. The comparison results in consideration whether some new knowledge concerning the correctness, consistency, completness, and unambiguity could be made.

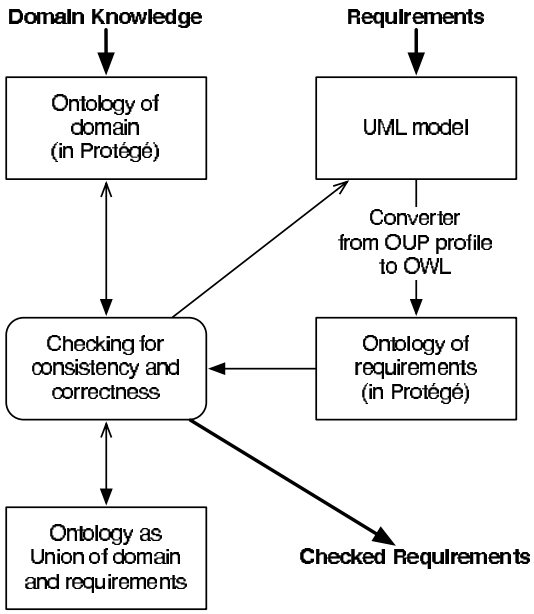

Figure 2. The component for consistency checking

There are some tools available. The UMLtoOWL tool by Gasevic [8] converts UML model description in extended Ontology UML Profile (OUP) using the XML Metadata Interchange (XMI) format to The Web Ontology Language (OWL) ontologies. The tool is implemented using eXtensible Stylesheet Language Transformation (XSLT).

We have used the Eclipse Framework and the ATL [18] Use Case UML2OWL by Hillairet [11]. He implemented a transformation according to the ODM specification. It consists of two separate ATL transformations. The first transformation UML2OWL takes an UML model as input and produces an ontology as OWL metamodel. The second transformation is an XML extractor that generates an XML document according to the OWL/XML specification by the W3C. We have extended Hillairets scripts to fit the UML models of TESSI and added support for SWRL contraints. These constraints are converted to SWRL/XML syntax to fit inside the OWL. This is done by an ANTRL parser and compiler that can convert SWRL rules in informal syntax entered in TESSI into the correct OWL/SWRL syntax. The use of SWRL rules provides us further posibilities for checking our model.

5.2. Using Pellet for Checking Ontology

Pellet is a tool that allows ontology debugging in the sense that it indicates the relation between unsatisfiable concepts or axioms that cause an inconsistency. We use it to check whether the requirements problem ontology subsumes the domain ontology. Because our problem ontology is generated from the UML model by the convertor ATL,

there are no problems to be expected in the structure of the problem ontology because the UML model has been built under respecting rules for well-formed UML model.

The OWL files generated in the previous step can be loaded into Protégé. From there they can be transferred to a reasoner using the DIG description logic reasoner interface. The DIG interface is an emerging standard for providing access to description-logic reasoning via an HTTP-based interface to a separate reasoning process. Current releases of Protégé already include the Pellet reasoner, since it is robust and scalable, and is available under an open-source license.

5.3. Using Jess for Reasoning in Ontology

To find inconsistencies in ontology rules we need an inference machine. We used the Jess rule engine [6]. Jess was inspired by the CLIPS expert shell system and adds additional access to all the powerful Java APIs for networking, graphics, database access, and so on. Jess can be used free of charge for educational purposes. Because Protégé and Jess are implemented in Java, we can run them together in a single Java virtual machine. This approach lets us use Jess as an interactive tool for manipulating Protégé ontologies and knowledge bases.

Protégé offers two ways to communicate with Jess. The first one is the plugin JessTab, which provides access to the Jess console and supports manual mapping of OWL facts into Jess and back. We used the second plugin SWRLTab. It is a development environment for SWRL rules in Protégé and supports automatical conversion of rules, classes, properties and individuals to Jess. From there you can control the Jess console and look up the outputs. This is done by a plugin for SWRLTab called SWRLJessTab, which contains a SWRL to Jess bridge and a Jess to Java bridge. This allows the user to add additonal functions to their SWRL rules by defining the corresponding functions as Java code and use them inside Protégé. SWRLTab lets you also insert the inferred axioms back into your ontology. This way it is possible to use complex rules to infer new knowledge.

5.4. Interaction of the used tools

All these described tools are put together during the requirement analysis. Figure 3 shows how this is done. Starting with the textual description and an ontology describing the domain the analyst can use TESSI to create a model of the planned system. This model can also contain constraints which will be compiled into SWRL rules by an ANTLR parser. The rest of the model will be transformed into an UML model [31], which will later be converted into an ontology. The ontology is merged with the SWRL rules and can then be opened in Protégé. From there the analyst can check the model consistency with Pellet and validate the rules with SWRLTab and Jess. The knowledge gained will then be used to make corrections to the TESSI model.

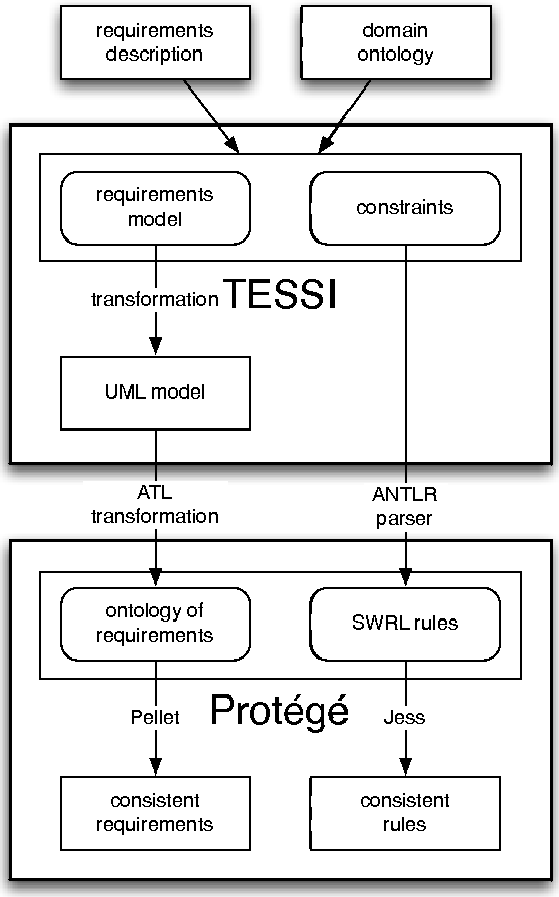

Figure 3. Interaction of the used tools

6. Experiments

For experiments, we used a requirements specification of a library [23]. The text describes functional requirements for a library management system. It contains aspects of media and user management, describing several special use cases and state machines for selected classes. The text was developed to show the posibilities TESSI provides for requirement analysis.

6.1. Checking with rules

As an example of checking rules we have the following case. There is a relation “borrow” between Librarian and User. But if we model Librarian as a subset of class User, because a librarian may also borrow books, the Librarian (as an instance) could borrow a book himself. This is not what we want. Usually, we do not allow that a clerk in a bank

can give a loan to himself, we do not want that a manager decides about his salary etc. The solution is that we do not allow some relations to be reflexive in the domain ontology, e.g. the relation "borrow". Any problem ontology that does not contain the condition that a librarian must not borrow a book to himself will be found to be inconsistent to the domain ontology.

This example can be checked in TESSI by modelling the two classes User and Librarian. We decide that a Librarian is a specialization of an User with additional possibilities to manage the library. Then we define an association between these two and name the direction from Librarian to User borrowsTo. After that we can use SWRL-Rules to describe the desired behavior. The first rule we need will set the relation between every possible Librarian and User pair:

$$\mathrm{Librarian}(?x) \wedge \mathrm{User}(?y) \rightarrow \mathrm{borrowsTo}(?x, ?y)$$

The second rule will be used to check if any of the librarians is able to borrow a book to himself:

$$\mathrm{borrowsTo}(?x, ?y) \wedge \mathrm{sameAs}(?x, ?y) \rightarrow \mathrm{error}(?x, \text{"self"})$$

Now we can create an UML model based on our example and then generate an ontologie with this content. The ontology will then be loaded into Protégé.

The Protégé plugin SWRLTab offers several ways to work with SWRL rules. It also allows us to communicate with the Jess rule engine. Using this plugin we can transform the knowledge of the ontology in facts for Jess. Running Jess will then cause new facts to be inferred.

In our case it will set up the borrowsTo relationship for all Users and Librarians and then test for Librarians that borrow to theirselves. The Inferred Axioms window in Protégé will then list all possible errors and we can use this information to make correnctions to the model in TESSI. In this case we can remove the subclass from User and after a further test Jess will get no errors.

6.2. Checking with restrictions

The next example will show the posibility to check restrictions. In our library a user can borrow books or reserve them if they are not available. In order to limit users to a fixed amount of reservations the reserve relation should be restricted.

In TESSI these conditions can be modeled with associations. Therefor we use the artifact dialog for associations to create a new Instance at the corresponding position in the requirements text. We set User and MediumInstance as association ends. The direction from User to MediumInstance will be labeled with reservedMedia an gets the cardinality 0 to n, in this example n is set to 3. Both classes User and MediumInstance must have set some equivalents in the domain ontology to access the corresponding individuals later. To provide some test data we need to add a constraint that fills the reservedMedia relation:

$$\mathrm{User}(?x) \wedge \mathrm{MediaInstance}(?y) \rightarrow \mathrm{reservedMedia}(?x, ?y)$$

After converting the model to UML and to an ontology we use the SWRLTab to infer the new axioms and then use the Jess to OWL button to include the new knowledge into our ontology. Afterwards we can check the results on the individuals tab in Protégé. It will show red borders around properties which do not meet the defined restrictions.

Based on these observations either the restrictions must be corrected or the test data is wrong and the constraint for filling it must be adopted.

7. Conclusion

One of the problems that may occur is that the restriction rules of requirements (called constraints) are described in OCL (Object Constraint Language) which is stronger than SWRL. As we will describe below, description logics have different expresiveness. The reason is that the computational complexity of the reasoning, i.e. of the decision whether the system is correct and consistent, may explode and we never obtain the result guarenteed if the expressivenes of the used description logic is too high.

Another problem is the necessarily use of individuals to process SWRL rules. This requires to add several individuals of every class to the domain ontology whithout knowing what rules will later be modeled in TESSI. It also requires to have some meaningful properties set to these objects. Otherwise it will not be possible to validate the model with SWRL rules.

SWRL also offers only limited possibilities to express rules. The formulas are based on first oder logic but can only contain conjunctions of atomic formulas. There is no support for quantiviers or more complex terms. SWRL also can't express negations, which requires the user to create formulas on a special way and limits the expressiveness of SWRL rules.

Ontology research has primarily focused on the act of engineering ontologies or it has been explored for use in domains other than requirements elicitation, specification, checking, and validation. Using ontologies supports consistency which is critical to the requirements engineering process. Consistent understanding of the domain of discourse reduces ambiguity and lessens the impact of contextual differences between participants.

In this paper we focused on consistency but there are also further properties that can be checked using ontologies. This is a motivation for our further research.

References

[1] Boehm B., In H.: Identifying Quality Requirements Conflicts. IEEE Software, pp. 25-35, March 1996, 1996.

[2] Carreno, R.S., et al.: An ontology-based approach to knowledge acquisition from text. Cuadernos de Filologia Inglesa, 9(1), pp. 191-212 (in English), 2000.

[3] CHAOS Report. The Standish Group, 1995.

[4] Christopherson, L.L.: Use of an ontology-based note-taking tool to improve communication between analysts and their clients. A Masters Paper for the M.S. in I.S.degree, University of North Carolina, November, 2005.

[5] Easterbrook, S., Callahan, J., and Wiels, V.: V & V Through Inconsistency Tracking and Analysis. Proceedings of International Workshop on Software Specification and Design, Kyoto, 1998.

[6] Eriksson, H.: Using JessTab to integrate Protege and Jess. Intelligent Systems, Volume 18, Issue 2, pp. 43- 50, IEEE Mar-Apr 2003.

[7] Falkovych, K.: Ontology Extraction from UML Diagrams. Master's thesis, Vrije Universiteit Amsterdam, 2002.

[8] http://www.sfu.ca/ dgasevic/projects/UMLtoOWL

[9] Gasevic, D., Djurevic, D., Devedzic, V.: Model Driven Architecture and Ontology Development. Springer, 2006.

[10] Guarino, N.: Understanding, building, and using ontologies. International Journal of Human-Computer Studies, 46(2/3), pp. 293-310, 1997.

[11] Guillaume Hillairet: ATL Use Case - ODM Implementation (Bridging UML and OWL) http://www.eclipse.org/m2m/atl/usecases/ODMImplementation.

[12] Horridge, M., et al.: A Practical Guide to Building OWL Ontologies Using the Protg-OWL Plugin and CO-ODE Tools, Edition 1.0. http://protege.stanford.edu/doc/users.html.

[13] Hunter, A., Nuseibeh, B.: Managing Inconsistent Specifications: Reasoning, Analysis and Action. ACM Transactions on Software Engineering and Methodology, Vol. 7, No. 4, pp. 335-367, 1998.

[14] Janetzko, R.: Applying ontology for checking of requirements specification. M.Sc. Thesis, Faculty of COmputer Science, TU Chemnitz, 2009. (In German)

[15] Jess 7.1 manual.
Sandia National Laboratories.
http://www.jessrules.com/jess/docs/index.shtml,2007.{OMG}
http://www.omg.org/techprocess/meetings/schedule/
UML_2.0_Superstructure_FTF.html

[16] Jiang, D., Zhang, S., Wang, Y.: Towards a formalized ontology-based requirements model. Journal of Shanghai Jiaotong University (Science), 10(1), pp. 34-39, 2005.

[17] Jin, Z.: Ontology-Based Requirements Elicitation. Journal of Computers, 23(5), pp. 486-492, 2003.

[18] Jouault, F., Allilaire, F., Bezivin, J., Kurtev, I.: ATL: A model transformation tool. Science of Computer Programming, Vol. 72, No. 1-2. pp. 31-39, June 2008.

[19] Kaiya, H., Saeki, M.: Using Domain Ontology as Domain Knowledge for Requirements Elicitation. In: Proceedings of 14th IEEE International Requirements Engineering Conference, Minnesota, pp. 186-195, 2006.

[20] Kroha, P., Strauss, M.: Requirements Specification Iteratively Combined with Reverse Engineering. In: Plasil,F., Jeffery,K. (Eds.), SOFSEM'97: Theory and Practice of Informatics. Milovy, November 1997, Lecture Notes in Computer Science, No. 1338, Springer, 1997.

[21] Kroha, P.: Preprocessing of Requirements Specification. In: Ibrahim, M., King, J., Revell, N. (Eds.): Proceedings of the 11th International Conference Database and Expert Systems Applications DEXA 2000, London, Lecture Notes in Computer Science, No. 1873, Springer, 2000.

[22] Kroha, P., Labra, J.: Using Semantic Web Technology in Requirements Specifications. Research Report Chemnitzer Informatik Berichte CSR-08-02, ISSN 0947-5125, TU Chemnitz, 2008.

[23] Kroha, P., Rink, M.: Text Generation for Requirements Validation. In: Filipe, J. and Cordeiro, J. (Eds.): Proceedings of ICEIS'2009, Milano, Lecture Notes in Business Information Systems 24, pp. 467-478, Springer, 2009.

[24] Li, Z., Wang, Z., Zhang, A., Xu, Y.: The Domain Ontology and Domain Rules Based Requirements Model Checking. International Journal of Software Engineering and Its Applications, Vol. 1, No. 1, July, 2007.

[25] Neches, R., Fikes, R.E., Finin, T.: Enabling technology for knowledge sharing. AI Magazine, 12 (3), pp. 36-56, 1991.

[26] Nuseibeh, B., Easterbrook, S., Russo, A.: Leveraging Inconsistency in Software Development. IEEE Computer, April 2000.

[27] http://protege.stanford.edu/overview/index.html

[28] Robinson, W., Fickas, S.: Supporting Multiple Perspective Requirements Engineering. In: Proceedings of the 1st International Conference on Requirements Engineering (ICRE 94), IEEE Computer Society Press, pp.206-215, 1994.

[29] Spanoudakis, G., Zisman, A.: Inconsistency Management in Software Engineering: Survey and Open Research Issues. In: Chang, S.K. (ed.): Handbook of Software Engineering and Knowledge Engineering, World Scientific Publishing Co., pp. 329-380, 2001.

[30] Uschold, M., King, M., Moralee, S., Zorgios, Y.: The Enterprise Ontology. Knowledge Engineering Review, 13(1), pp. 31-89, 1998.

[31] Weidauer, J.: Implementation of a Component for Transformation of an OWL Models into UML 2. M.Cs. Thesis, Faculty of Computer Science, TU Chemnitz, 2008. (In German)

[32] Zhu, X., Zhi, Jin: Inconsistency Measurement of Software Requirements Specifications an Ontology-Based Approach. In: Proceedings of the 10th IEEE International Conference on engineering of Complex Computer Systems, 2005.