Ontologies in Checking for Inconsistency of Requirements Specification

Article · October 2009

DOI: 10.1109/SEMAPRO.2009.11

CITATIONS
16

READS
186

3 authors:

Petr Kroha
Technische Universität Chemnitz
70 PUBLICATIONS **221** CITATIONS
SEE PROFILE

Robert Janetzko
Technische Universität Chemnitz
2 PUBLICATIONS **17** CITATIONS
SEE PROFILE

Jose Emilio Labra Gayo
University of Oviedo
159 PUBLICATIONS **917** CITATIONS
SEE PROFILE

2009 Third International Conference on Advances in Semantic Processing

Ontologies in Checking for Inconsistency of Requirements Specification

Kroha, P., Janetzko, R.
TU Chemnitz, Faculty of Computer Science, Chemnitz, Germany
e-mail: 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
e-mail: jelabra@gmail.com

***Abstract*—In this paper, we investigate how ontologies developed for use in Semantic Web technology could be used in checking the consistency of requirements specifications. Our approach's main idea and original contribution is that we use reasoning which is a part of ontology. In the first step, we transform the static part of the UML model and its constraints into a problem ontology and try to discover contradictions using ontology reasoning. The contradictions that have been found indicate inconsistencies. In the second step, we try to discover contradictions between the problem ontology coming from the UML model, that represents the requirements, and the domain ontology, that represents the domain knowledge available in the software company. However, a limitation of this approach is: We cannot check the behavioral consistency because it is not possible to represent the dynamic part of the UML model in an ontology. This paper describes not only a concept but also the implementation and illustrating examples.**

***Keywords*-requirements; consistency; checking; ontology; reasoning**

I. 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 results in the development of software parts that do not fit to the real needs of the user and must be reworked later causing 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). Then, they will note the requirements they have 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, the analyst specifies a UML model (Unified Modeling Language) [32] based on the requirements description he has written himself. However, users and domain experts cannot validate the UML model as most of them do not understand (semi-)formal languages such as UML. Misunderstandings between analysts and users are very common and bring projects over budget.

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

The motivation of our project and its goal can be characterized with the following questions:

- "Is it possible to improve requirements specifications using ontologies?"
- "What part of it can be improved?"
- "How difficult is the implementation?"
- "What we can gain?"

In this paper, we introduce our concept that uses ontology reasoning, we present the implementation and respond to the questions mentioned above in the conclusion.

We have designed and implemented the presented approach as an ontology-based component in our requirements specification tool TESSI [20], [21], [23] that helps an analyst not only to write a UML model but also to improve it and to reduce misunderstandings.

In this paper, we focus on the problem of consistency checking only because of the limited space, i.e., we will not discuss the problems of how to construct a domain ontology, how to gather requirements, etc..

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 tool TESSI. In Section 5, the implementation of the component that represent our approachs is given. In Section 6, we present the results of experiments, Section 7 discusses limitations in constraint languages, and the Section 8 contains conclusions and future work.

II. Related work

Using ontologies in the requirements engineering process is clearly not a new idea. An ontology-based approach to knowledge acquisition is discussed in [16], in [18],

978-0-7695-3833-4/09 $26.00 © 2009 IEEE
DOI 10.1109/SEMAPRO.2009.11

and in [3]. In [33], they have constructed the Enterprise Ontology to assist developers in having an enterprise-wide view of an organisation. Such an ontology could be used as our predefined domain ontology. The approach in [16] is intended to automate both, interactions with users and the development of application models. The ontologies used by [29] in their Oz system are domain models which prescribe detailed hierarchies of domain objects and the relationships between them.

Formal models for ontology in requirements are described in different papers, e.g., in [15]. In [24], the UML model is enhanced with a formal semantics and used for working with a use-case model and class diagrams. In [25], an ontology model for requirements elicitation for domain knowledge and domain rules checking is presented. In [34], the inconsistency measurement is discussed. The ontology used by the QARCC system [1] is a decomposition taxonomy of software system quality attributes. In [25], a formal model of requirements elecitation that contains domain ontology checking is discused.

The concept used for consistency checking in the 1990s based on the relationships between viewpoints and their problematic inter-depencies [4]. An overview about inconsistencies is given in [5], in [26], and lately in [30]. However, there is not an approach applying ontology in the sense of our way.

Another approach is a concept based on the semantics of dependency as described by the UML specification. It will be used to describe the behavioral part of the UML model [19]. Other research checking behavioral consistency is [31] or [9]. They are working with the dynamic model.

III. Why use Ontology for Checking Requirements Specifications

Ontologies 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 (as problem ontology in our text). This part of ontology corresponds to a static part (or structural part) of the UML model.
- Reasoning part - a logical theory that constrains the intended model containing:
  - integrity rules of the domain model representing the domain knowledge,
  - derivation rules and constraint rules of the problem model.

  Reasoning in ontologies adds the inferential capabilities that are not present in taxonomies which have beenn used for modeling so far. It makes possible to search for contradictions that indicate inconsistencies. The UML model includes a constraint part but it has no mechanism for indicating contradictions in constraints. When many UML models are considered, the resulting UML model has no mechanism for indicating a discrepance in the class hierarchy.

When we transform the requirements specifications into a UML model (using TESSI in our case), we either have to believe that the analyst understands the user's needs perfectly and is perfect in UML model building or we do not believe it and to try to verify the UML model. Ontology engineering offers a mechanism to discover contradictions in constraints and in class hierarchies. To use it, it is necessary to transform the UML model into an ontology which we call a problem ontology.

In a specialized software house, developing for example information systems for banking, a domain ontology can be constructed that represents the domain knowledge. Such a domain ontology can be merged with the problem ontology of the current project and the resulting ontology can be tested for contradictions as described above.

The contradictions that have been found indicate inconsistencies in the requirements specifications.

The fact that it is not possible to test the dynamic part of the UML model (the behavior part) is a disadvantage of this approach. However, the methods of UML behavioral consistency checking (in [31]) depend on formal semantics methods and they are very complex.

IV. The tool TESSI – Architecture and Dataflow

In this section, we will briefly introduce our tool TESSI [20], [21], [23] to bring the context information that is necessary to see the whole workflow of how to do requirements processing with our tool.

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 is used and tested.

We developed a textual refinement of the requirements specifications which can be called requirements description. Working with it, the analyst uses the grammatical inspection to specify the roles of words in the text in the sense of object-oriented analysis. During this process—shown in Fig. 1—a UML model will be built driven by the analyst's decisions. Based on this UML model a new, model-derived requirements description in textual form will automatically be generated [23] that describes the analyst's understanding of the problem. Now, the user can read it, understand it and validate it. The analyst will use his/her comments for a new version of the requirements description. The process repeats

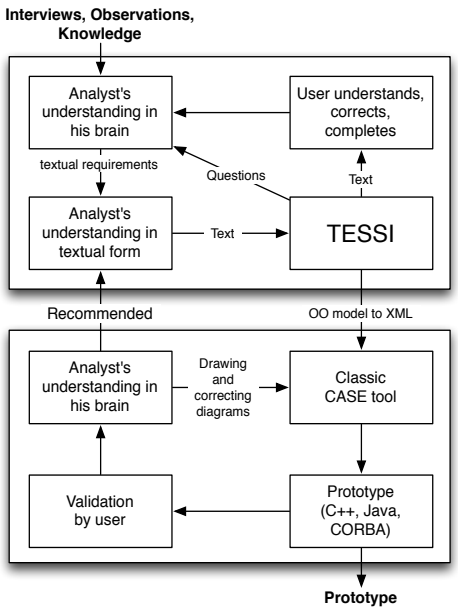

Figure 1. Architecture and dataflow of our tool TESSI

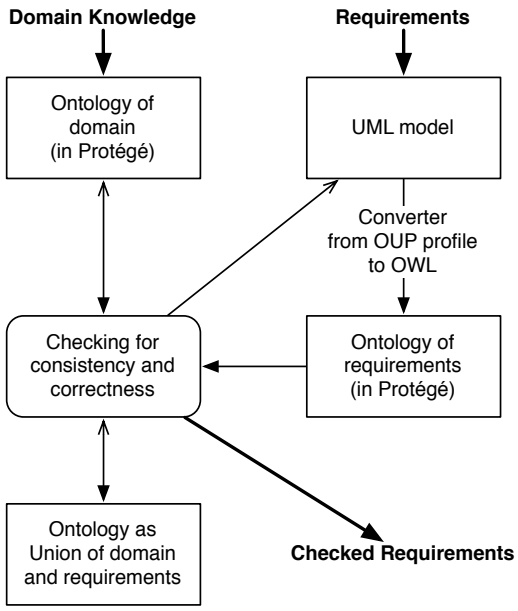

Figure 2. The conceptual model of the consistency checking component

until there is a consensus between the analyst and the user. This does not have to mean that the requirements description is perfect, but some mistakes and misunderstandings should have been removed.

In this paper, we enhance the approach described above and present how the UML model is used to check consistency. A model of our concept is shown in Fig. 2. Using the experiences given in [25], we describe the domain ontology in Protégé [28] and apply ontology reasoning (e.g., the inference engine in Pellet [27] – for checking classes) first for domain ontology checking, then for requirements problem ontology checking, and at last for checking whether the requirements problem ontology subsumes the domain ontology.

The steps of the requirements processing using ontologies are the following:

- building a domain ontology using Protégé, i.e., domain ontology description 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 using convertor ATL,
- checking the problem ontology for its consistency,
- merging the problem ontology with the domain ontology and checking them for consistency
- 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.

V. Implementation

The component of TESSI containing the ontology-based consistency checking of requirements specification has been implemented in [13]. Our goal had been to convert the UML model—obtained with the help of grammatical inspection—from the textual requirements into a corresponding problem ontology model that can be verified and compared with the domain ontology model to find contradictions.

As we already mentioned above we needed to implement:

- Converting UML model into a problem ontology model - we have used the Eclipse Framework and the ATL [17] Use Case UML2OWL by Hillairet [10]. We have

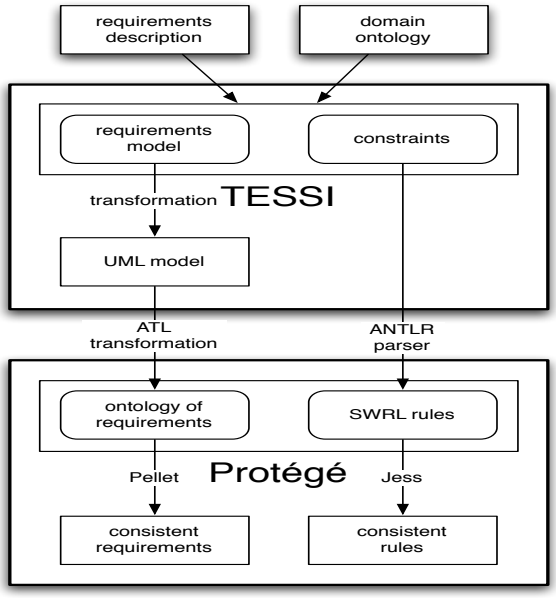

Figure 3. Interaction of the tools

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.

- Checking ontology class hierarchy - Current releases of Protégé already include the Pellet reasoner. We use it to check whether the 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.
- Checking consistency of ontology rules - To find inconsistencies in ontology rules we need an inference machine. We used the Jess rule engine [6]. Because Protégé and Jess are implemented in Java, we can run them together in a single Java virtual machine. Protégé offers two ways to communicate with Jess. The first one is the plugin JessTab, 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.

All these tools described above (Fig. 3) work together during the requirements analysis. Starting with the textual description and a domain ontology, the analyst (not the end user of the proposed system) can use TESSI to create a UML model of the planned system, which will later be converted into a problem ontology. The problem ontology is merged with the SWRL rules describing constraints, with domain ontology and can then be opened in Protégé. From there the analyst can check the problem ontology consistency, i.e., the UML model consistency, with Pellet and validate the rules with SWRLTab and Jess. If some contradictions are found by Pellet (in class structures) or by Jess (in rules) the knowledge gained will then be used to make corrections to the TESSI model.

VI. Experiments

For experiments, we used the requirements specification of a library [23] that describes the functional requirements for a library management system.

A. Checking with rules

Let us consider this example for checking rules: There is a relation "lendsTo" between classes Librarian and User. But if we model the class Librarian (additionally feature of lending books) as a subset of the class User, the librarian (as an instance) could lend a book to himself. This is usually 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 a manager decide 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 "lendsTo". Any problem ontology that does not contain the condition that a librarian must not lend 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 a User with additional possibilities to manage the library. Then we define an association between these two and name the direction from Librarian to User "lendsTo". 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:

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

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

$$\mathrm{lendsTo}(?x, ?y) \wedge \mathrm{sameAs}(?x, ?y) \rightarrow \mathrm{error}(?x, \text{“self”})$$

Now, we can create a UML model based on our example and then generate an ontologie with this content and load it into Protégé. It will set up the "lendsTo" relationship for all Users and Librarians and then test for Librarians that lend to themselves. The Inferred Axioms window in Protégé will then list all possible errors so we can use this information to make correnctions to the UML model. In this case, we can remove the subclass from User. During a further test, Jess will get no errors.

B. 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 our modeling tool we set User and MediumInstance as association ends. The direction from User to MediumInstance will be labeled with reservedMedia and has 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 an ontology we use the SWRLTab to infer the new axioms and then use the Jess to include the new knowledge into our ontology. Afterwards we can check the results on the individuals tab in Protégé. It will indicate that the defined restrictions are not fulfilled. 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.

VII. Problem of constraints description

One of the problems that occurs is that the constraints of requirements are described in OCL (Object Constraint Language) which is stronger than SWRL (Semantic Web Rule Language). 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. Hence, we can use only the subset of OCL that corresponds to SWRL.

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 quantifiers or 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.

Another problem is the necessity to use 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 meaningful properties set to these objects. Otherwise it will not be possible to validate the model with SWRL rules.

On the other side, there are hundreds of constraints in most UML models which we can describe using SWRL, e.g., value ranges of attributes and other not very complicated conditions. If we cannot describe a constraint in SWRL, then the only consequence is that our prototype cannot use it for checking.

VIII. Conclusion and Future Work

We have shown that using ontologies supports consistency checking which is critical to the process of requirements engineering. Reasoning as a part of ontology discovers contradictions in the class hierarchies and the constraints of UML models and thereby helps in indicating inconsistencies.

In the introduction of this paper we stated the motivated questions. Now, we can answer.

Lessons learnt:

- "Is it possible to improve requirements specifications by using ontologies?" Yes, it is possible. We presented a concept and a tool that finds contradictions and inconsistencies in UML models.
- "What part of it can be improved?" Using ontologies, we can improve only those parts of the UML model that can be transformed into ontologies. These are the structural part (static part of the model - class hierarchy) and the constraint part of the model. We cannot improve the behavioral part (the dynamic part of the model - state machine diagrams, sequence diagrams, etc.).
- "How difficut is the implementation?" The implementation took about four months for a single person. We were able combine many public available tools. At the very moment, we are testing the first version. Our goal was to have a simple prototype as soon as possible. We will continue to develop improved versions.
- "What we can gain?" Using our approach we can gain indications about contradictions in class hierarchy and in constraints in the problem ontology and indications about contradictions between the problem ontology and the domain ontology.

In this case, we used the "depth-first" method of research, i.e., we were trying to get a running prototype as soon as possible. Many interesting theoretical problems and questions appeared during our project. Because of the limited space in this paper, we cannot discuss them all. In further research, we will try to solve them. We will also try to use our tool for real projects to obtain experiences and new stimuli.

References

[1] B. Boehm and H. In, "Identifying quality requirements conflicts," IEEE Software, pp. 25-35, March 1996.

[2] CHAOS Report. The Standish Group, 1995. http://www.projectsmart.co.uk/docs/chaos-report.pdf

[3] L.L. Christopherson, "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.

[4] S. Easterbrook and B. Nuseibeh, "Using ViewPoints for Inconsistency Management," IEEE Software Engineering Journal, November 1995.

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

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

[7] K. Falkovych, “Ontology Extraction from UML Diagrams,” Master’s thesis, Vrije Universiteit Amsterdam, 2002.

[8] D. Gasevic, D. Djurevic, and V. Devedzic, “Model Driven Architecture and Ontology Development,” Springer, 2006.

[9] C.L. Heitmeyer, R.D. Jeffords, and B.G. Labaw: “Automated COnsistency Checking of Requirements Specifications,” ACM Transactions on Software Engineering and Methodology, Vol. 5, No. 3, pp. 231-261, July 1996.

[10] G. Hillairet, “ATL Use Case - ODM Implementation (Bridging UML and OWL),” http://www.eclipse.org/m2m/atl/usecases/ODMImplementation.

[11] M. Horridge, H. Knublauch, A. Rector, R. Stevens, and C. Wroe, ”A Practical Guide to Building OWL Ontologies - Using the Protege-OWL Plugin and CO-ODE Tools,” Edition 1.0., http://protege.stanford.edu/doc/users.html.

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

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

[14] E. Friedman-Hill, “Jess in Action - Rule-based Systems in Java,” Manning, Greenwich, 2003.

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

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

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

[18] H. Kaiya and M. Saeki, “Using domain ontology as Domain Knowledge for Requirements Elicitation,” Proceedings of 14th IEEE International Requirements Engineering Conference, Minnesota, pp. 186-195, 2006.

[19] S. Kremer-Davidson and Y. Shaham-Gafnil, “UML 2.0 Model Consistency - The Rule of Explicit and Implicit Usage Dependencies,” UML 2004 Modeling Languages and Applications, 3rd Workshop on Consistency Problems in UML-based Software Development: Understanding and Usage of Dependency Relationships, Lisboa, 2004.

[20] P. Kroha and M. Strauss, “Requirements Specification Iteratively Combined with Reverse Engineering,” SOFSEM’97: Theory and Practice of Informatics. Milovy, November 1997, Lecture Notes in Computer Science, No. 1338, Springer, 1997.

[21] P. Kroha, “Preprocessing of Requirements Specification,” Proceedings of the 11th International Conference Database and Expert Systems Applications DEXA 2000, London, Lecture Notes in Computer Science, No. 1873, Springer, 2000.

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

[23] P. Kroha and M. Rink, “Text Generation for Requirements Validation,” Proceedings of ICEIS’2009, Milano, Lecture Notes in Business Information Systems 24, pp. 467-478, Springer, 2009.

[24] X. Li, Z. Liu, and J. He, “Consistency checking of UML requirements,” Proceedings of the 10th IEEE International Conference on Engineering of COmplex Computer Systems ICECC, 2005.

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

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

[27] http://pellet.owldl.com

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

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

[30] G. Spanoudakis and A. Zisman, “Inconsistency Management in Software Engineering: Survey and Open Research Issues,” Handbook of Software Engineering and Knowledge Engineering, World Scientific Publishing Co., pp. 329-380, 2001.

[31] Y. Thierry-Mieg and L. Hilla, “UML behavioral consistency checking using instatiable Petri nets,” Journal Innovations in Systems and Software Engineering, Springer, Vol. 4, No. 3, 2008.

[32] http://www.omg.org/spec/UML/2.0/

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

[34] X. Zhu and J. Zhi, “Inconsistency Measurement of Software Requirements Specifications an Ontology-Based Approach,” Proceedings of the 10th IEEE International Conference on engineering of Complex Computer Systems, 2005.