Using Text Semantic Similarity Approach to Check the Consistency of UML

Yasser Kotb* and Takuya Katayama**

Japan Advanced Institute of Science and Technology
School of Information Science
Asahidai 1-1, Nomi, 923-1292, Ishikawa, Japan

IS-RR-2006-013
September 7, 2006

* Corresponding author: kotb@jaist.ac.jp.
** Corresponding author: katayama@jaist.ac.jp

Using Text Semantic Similarity Approach to Check the Consistency of UML[1]

Yasser Kotb[2] and Takuya Katayama[3]

Japan Advanced Institute of Science and Technology
School of Information Science
Asahidai 1-1, Nomi, 923-1292, Ishikawa, Japan

Abstract

It is an important and stimulating issue to discover the inconsistency and incompleteness of the large software system through the UML diagrams. However, the use of temporal logic and model checking techniques to address the problem of UML consistency for developing software systems has received much attention. It is still not applicable and hard mission to specify the different consistency issues of UML. In this paper, we address this problem. We investigate the use of a recent natural language processing technique called *Text Semantic Similarity*, or *Textual Entailment*, to improve the consistency and the completeness among various UML diagram.

Keywords: UML Language Engineering, Text Semantic Similarity, Consistency, Verification.

1 Introduction

The Unified Modeling Language UML [16] is becoming the de-facto notation for software engineering projects. Software systems are described using multiple views. These views are partially overlapping, e.g. class diagrams for the static structure and state charts for the behavior of the system. This separation of concerns on the one hand reduces the complexity of the overall specification, but on the other hand the increasing number of notations very often leads to a wide range of inconsistencies and incompleteness. For example, syntactical inconsistencies violate the well-formedness of the models; behavior inconsistencies violate the compatibility between different diagrams or create inconsistencies during refinement of the diagrams. It is a common hypothesis that incompleteness and inconsistency allowed by UML are a source of high risk problems in the software development process. It

[1] This research is supported by JSPS (Japan Society for the Promotion of Science) and the Grant-in-Aid for JSPS Fellows.

[2] Email: kotb@jaist.ac.jp

[3] Email: katayama@jaist.ac.jp

is well known that errors introduced early in the development process are usually the most expensive to correct [10]. Therefore, work on how to check the consistency of software systems which model by UML diagrams is necessary and useful.

The problem of consistency within and between different UML modeling artifacts arises independency of the used methodology. Our aim in this paper is describing a novel approach for checking the consistencies and incompleteness across UML diagrams. The system model consists of a class diagram, a family of sequence diagrams, a family of state machines and system constraints will be checked through a new natural language processing approach. This approach is called *Text Semantic Similarity* (TSS) [4] or more recently *Textual Entailment* (TE) [5] [1]. The TSS task is defined as recognizing, given two text fragments, whether the meaning of one text can be inferred from the other. This application independent task is suggested as capturing major inferences about the variability of semantic expression which are commonly needed across multiple applications. For instance, there is an obvious similarity between the text segments "I own a car" and "I have an automobile". TSS has been used for relevance feedback and text classification [14], word sense disambiguation [9], and more recently for extractive summarization [15], and methods for automatic evaluation of machine translation or text summarization [13].

In this paper, we propose the use of the novel natural language processing approach, TSS, for checking large software systems, which is designed using the various UML diagrams. In this regard we investigate the typical approach to find the similarities between two text segments. As using a simple lexical matching method, and produce a similarity score based on the number of lexical units that occur in both input segments. Improvements to this simple method have considered stemming, stop-word removal, part-of-speech tagging, longest subsequence matching, as well as various weighting and normalization factors [15].

The main features of the approach presented here allow one to automatically detect differences and inconsistencies between various UML diagrams. For instance, use case diagrams, class diagrams, sequence diagrams etc. This represents the various views of the same system model. The key idea here is to measure the degree of similarity of the various vocabularies, which represent the same actors, classes and message names in different semantically equivalent texts. In this regard, we propose a complete framework to check the consistency of UML diagrams using the text semantic similarity techniques.

This paper is a part of an ongoing effort to design a complete system for checking the consistency and completeness of UML system model diagrams [8] through XMI representation.

The rest of the paper is organized in the following way. Section 2 gives a brief introduction to the consistency problem and the text semantic similarity approach and their measurement techniques. In Section 3, we illustrate our motivation example that will play role to clarify our ideas in next sections. Section 4 proposes our framework that is checking the consistency and completeness among various UML diagrams. We call it the UML Text Similarity Framework. Section 5 addresses the

different related research topics to our research. Lastly, Section 6, we draw attention for some conclusions and some idea about future works.

2 Background Review

In this section, we discuss the notion of consistency problem as well as brief review to text semantic similarity approach.

2.1 The Notion of Consistency Problems

The consistency problems can be addressed into two major viewpoints [6]. The first one is the situation where the consistency occurs and the other one is depending on the consistency conditions.

For the first viewpoint, the problem of consistency arises in two different cases. First, when the system is modeled from different modeling viewpoints. This type of consistency problem is called *horizontal consistency.* Second, when a model is evolved into another model, or by replacing one or more of its sub models, then it is desirable that the replaced sub model is a refinement of the previous sub model, in order to keep the overall model consistent. This type of consistency problem is called *vertical consistency.*

For the second viewpoint, a different categorization is obtained by looking at the conditions for the consistency problem. We can distinguish between two classes; *syntactic consistency* and *semantics consistency.* In general, consistency is the property that different sub models of a model can be integrated into a single model with a well-defined semantics and can thus be considered as a semantic property. In order to ensure consistency, a number of inconsistent models can already be detected by regarding their syntax which means the semantics property of consistency is checked syntactically. Syntactic (semantics) consistency ensures that a model is consistent with respect to the syntax (semantics) and is ensured by formulating consistency conditions on the syntax (semantics) of models.

2.2 Text Semantic Similarity Approach

Text semantic similarity or recently the textual entailment defines the task that requires to recognize, given two text fragments, whether the meaning of one text can be inferred (entailed) from another text. More concretely, textual entailment is defined as a directional relationship between pairs of text expressions, denoted by T - the entailing "Text", and H - the entailed "Hypothesis". We say that T *entails* H if the meaning of H can be inferred from the meaning of T, as would typically be interpreted by people. This somewhat informal definition is based on (and assumes) common human understanding of language as well as common background knowledge. It is similar in spirit to evaluation of applied tasks such as *Question Answering* (QA) and *Information Extraction* (IE), in which humans need to judge whether the target answer or relation can indeed be inferred from a given candidate text.

In fact, one of the fundamental characteristic of natural language is the variability of text semantic expression, where the same meaning can be expressed by, or inferred from, different texts. This natural language characteristic may be considered as the language ambiguity twin problem. Both can be combined to form the many-to-many mapping between language expressions and meanings during the language processing approaches. Many natural language processing applications, such as *QA*, *IE*, *multi-document summarization*, and *machine translation evaluation*, need a model for this variability characteristic in order to recognize that a particular target meaning can be inferred from different text variants. Although there are many different applications that are need similar models for text semantic variability. This problem has been addressed many times in a different application-oriented manners and method views that are evaluated by their impact on final application performance. For example, one of the earliest applications of text similarity is perhaps the vectorial model in *information retrieval* (IR), where the document most relevant to an input query is determined by ranking documents in a collection in reversed order of their similarity to the given query [15].

Overall, these various approaches become difficult to compare such various practical methods that were developed within different applications under the same framework conditions. Furthermore, researchers within one application area might not be aware of relevant methods that were developed in the context of another application. This leads to big challenge to build a clear framework that has clear generic task definitions and evaluations. Recently, there are two consecutive attempts to investigate such challenge: the first Recognizing Textual Entailment Challenge (15 June 2004 - 10 April 2005) [5] and the second Recognizing Textual Entailment Challenge (1 October 2005 - 10 April 2006) [1]. Both try to promote an abstract generic task that captures major semantic inference needs across applications. However, as in other evaluation tasks, these challenges give a new definition of textual entailment from operational view and corresponds to the judgment criteria given to the annotators who decide whether this relationship holds between a given pair of texts or not.

Recently there have been a few suggestions in the literature to regard entailment recognition for texts as an applied, empirically evaluated, task [5]. For example, a QA system has to identify texts that entail a hypothesized answer. In certain *Information Retrieval* (IR) queries the combination of semantic concepts and relations denoted by the query should be entailed from relevant retrieved documents. In IE entailment holds between different text variants that express the same target relation. In multi-document summarization a redundant sentence, to be omitted from the summary, should be entailed from other sentences in the summary. In machine translation evaluation a correct translation should be semantically equivalent to the gold standard translation, and thus both translations should entail each other. Therefore, it is expected that the textual entailment recognition is a suitable generic task for evaluating and comparing applied semantic inference models.

2.2.1 Measuring Text Semantic Similarity

Given two input text segments, we want to automatically derive a score that indicates their similarity at semantic level, thus going beyond the simple lexical matching methods traditionally used for this task [4] [3]. We should take into account the relations between words, as well as the role played by the various entities involved in the interactions described by each of the two texts, we take a first rough cut at this problem and attempt to model the semantic similarity of texts as a function of the semantic similarity of the component words. Corley and Mihalcea [3] do this by combining metrics of word-to-word similarity and language models into a formula that is a potentially good indicator of the semantic similarity of the two input texts. For a given pair of text segments T_i and T_j, they start by creating sets of open-class words, with a separate set created for nouns, verbs, adjectives, and adverbs. In addition, they also create a set for cardinals, since numbers can also play an important role in the understanding of a text. Next, they try to determine pairs of similar words across the sets corresponding to the same open-class in the two text segments.

There is a relatively large number of word-to-word similarity metrics that were previously proposed in the literature, ranging from distance-oriented measures computed on semantic networks, to metrics based on models of distributional similarity learned from large text collections [3]. In fact most of these metrics are defined between concepts, rather than words, but they can be easily turned into a word-to-word similarity metric by selecting for any given pair of words those two meanings that lead to the highest concept-to-concept similarity.

The lexical cohesion can be considered as semantic similarity between words. Similarity is computed by spreading activation or association on a semantic network constructed systematically from an English dictionary [17]. It is given by a set of associative relation shared by the people in a linguistic community. In this case, the similarity between words is a mapping σ : L $\times$ L $\rightarrow$[0, 1], where L is a set of words (or lexicon). The value of σ (w, w') increases with strength of semantic relation between w and w'.

3 Motivating Example

We illustrate the application of the text semantic similarity to check the consistency of UML diagrams with an example. This example shows a part of the structure viewpoint of educational system. In this example, we will restrict ourselves to structure modeling view for the proposed educational system framework. However, our approach can be extended directly in same manner to others dynamic modeling views. Figure 1 shows a case diagram for educational system. In this class diagram: Department, Student and Module are representing the main classes in the system. While the Science and Law are derived classes from the general class Department. Figure 2 shows a sequence model for the educational system composed of classes that are extracted from the above class diagrams. The flow of events of this educational system is represented by messages of sequence diagrams.

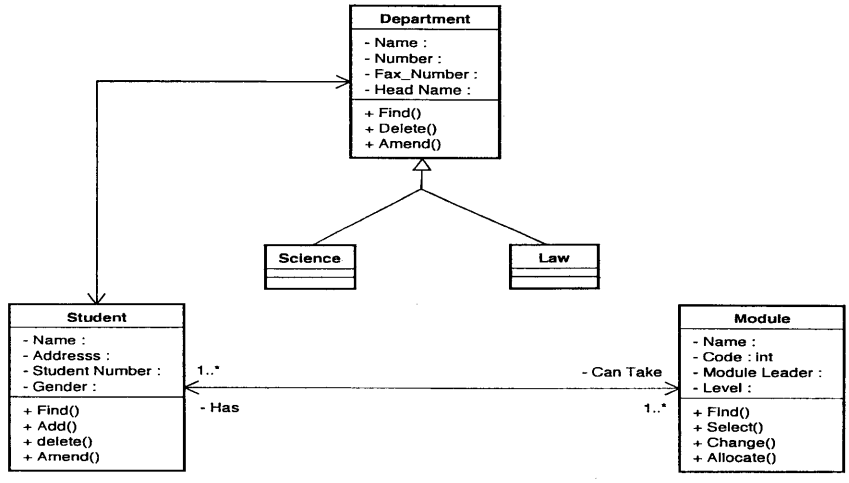

Fig. 1. Educational System class Diagram

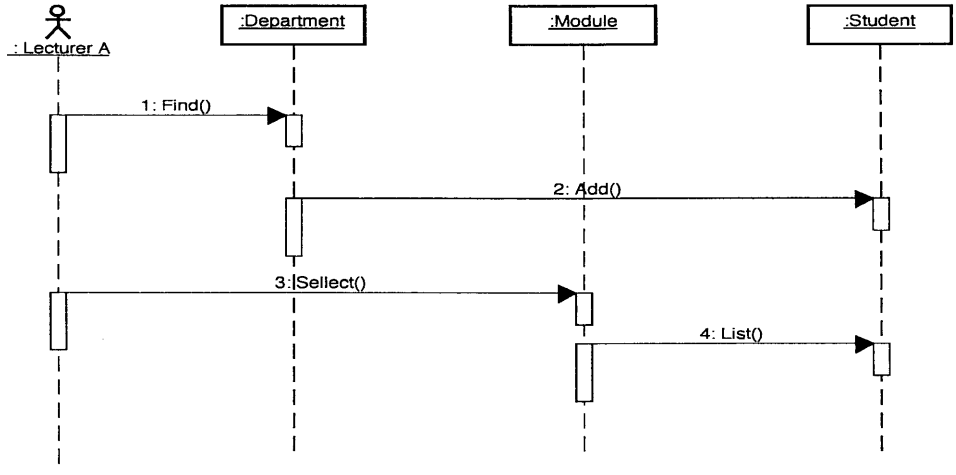

Fig. 2. Educational System Sequence Diagram (A)

As the complexity of software increases due to development of network techniques and the process of multimedia data, it has been essential to decompose the overall design of software framework. Moreover, most of the time, it is impossible to predict how software will have to evolve in some time. It might require more features, or some of its features have to be changed. Therefore, it is important to be able to decompose the softwares features as desired. By decomposing the software to be developed into different schemes according to the different functional domains will help us handle the overall software requirements complexity. This decomposition of the software is known by *separation of concerns*, it is a software engineering concept used to reduce the complexity of software. It refers to the ability to identify, encapsulate and manipulate only those parts of software that are relevant to a particular concept, goal or purpose [12]. Therefore, leading software companies each have a different system modeling groups. Each group is responsible to design the different aspect views of the real project. I.e. a group of designers are respon-

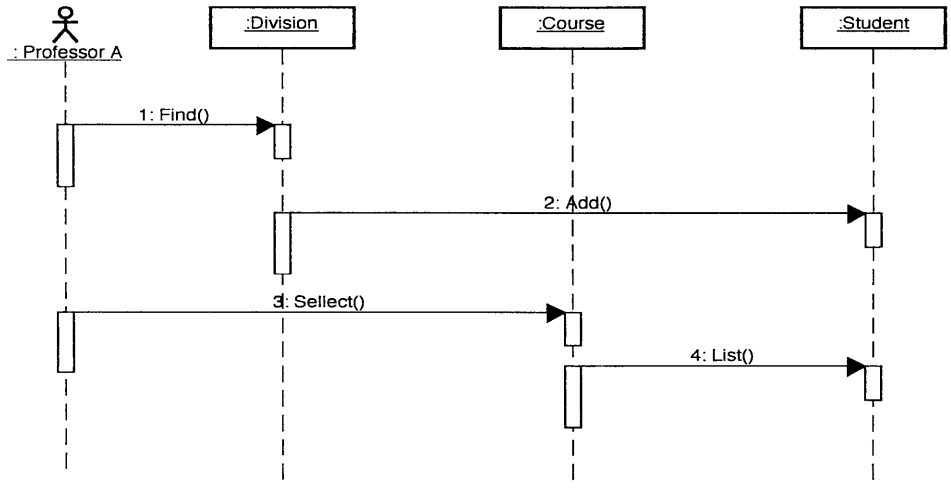

Fig. 3. Educational System Sequence Diagram (B)

sible for designing the use case description for the desired system software, while another group is responsible for model the class diagrams and so on. This leads to an inconsistency problem among these different groups, especially if each group used different vocabularies to represent there model view. For example, Figure 3 shows different sequence diagrams corresponding to our educational system example. Comparing Figure 2 and Figure 3, we can notice that both figures are almost same except that the actor name Lecturer in Figure 2 represented by the new actor name Professor in Figure 3. Again, the classes Department and Module in Figure 2 are represented by new classes named Division and Course in Figure 3 respectively.

At first glance it may seem that this is an easy task to discover such inconsistency manually. This is not true because really large software systems usually have huge vocabularies that lead to same meaning. There is one way to deal with this issue; the manger of the software system can build a main dictionary that contains all these vocabularies and their specified means. And each group now has to reference to this dictionary. Although, this approach seems easy and sufficient, it is impractical and difficult because it is manual manner that carries a lot of effort from all groups. This, of course, would increase the dependent comportment in the each design group. Moreover, establishing shared vocabularies and set of concepts among disparate teams is not applied in practice. In practice, the implementation of a system, whether from scratch or as an update to an existing solution, may be disconnected from the models. An interesting example of this is the growing number of organizations that outsource implementation and maintenance of their systems while maintaining control of the overall enterprise architecture [2]. This motivates us to find an automatic way two solve such inconsistency problems in real software system.

In the next section, we will present our novel approach to check such UML diagrams inconsistency problems by using this resent approach of natural language processing TST. As mentioned in previous section, TST approach is able to recognize if a given two text fragments, whether the meaning of one text can be inferred

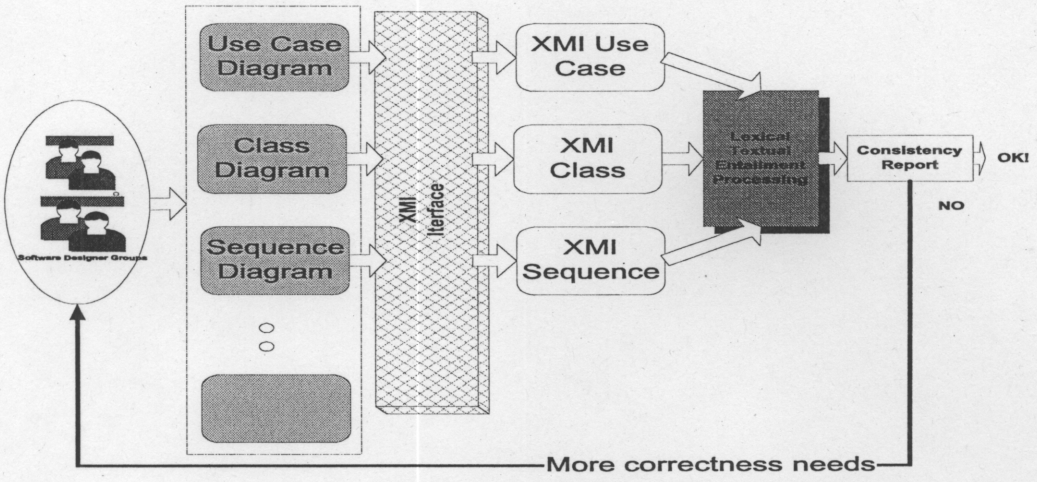

Fig. 4. UML Text Similarity Checker Framework

(entailed) from another text. This is the key idea here. We use this approach to check similarities between different vocabularies that are mentioned in various UML models. For simplicity, we restrict our approach to consider the texts that are simple words. Although in real application our approach can be applied if the text is a long sentence. Especially, in the real software, the different ambiguous vocabulary words are defined by associated comment notes. These notes, in general, are given by one or two paragraph of text.

4 UML Text Similarity Framework

Our proposed framework for checking the consistency and completeness among various UML diagrams, which is specified some desired software system, is sketched in Figure 4. The framework can be separated into three phases:

4.1 UML Model Design Phase

This phase will be our starting point (as shown in left part of the Figure 4). The software teams start to design the desired UML models for different parts of the software systems, which may contains inconsistencies and incompleteness problems. This phase can be done by using any of the existence UML Modeling tools. For example, IBM Rational rose (`http://www-306.ibm.com/software/rational/`) or ArgoUML (`http://argouml.tigris.org/`). In our approach, we have used the latter, ArgoUML.

4.2 UML-XMI Mapping Phase

This phase is responsible to convert between the UML model diagrams and their corresponding *XML Metadata Interchange* (XMI) documents [18]. XMI is a standard interchange mechanism used between various tools, repositories and middleware. The main purpose of XMI is to enable easy interchange of metadata between mod-

eling tools (based on the OMG UML) and between tools and metadata repositories (OMG MOF based) in distributed heterogeneous environments. XMI integrates three key industry standards: XML (*eXtensible Markup Language*), a W3C standard; UML (*Unified Modeling Language*), an OMG modeling standard; and MOF (*Meta Object Facility*) and OMG modeling and metadata repository standard. In our approach, we extract these XMI documents using the XMI export tool that is available inside ArgoUML system.

4.3 Text Similarity Checker Phase

This phase is the kernel part in our approach. It employs the text similarity manner in order to check the inconsistencies amongst the different UML diagrams through their corresponding XML documents. This is done as follows: first, we convert the given UML model diagrams to its equivalent XMI documents using the UML-XMI interface part mentioned above. Then, we extract the different vocabularies from each XMI document into different classes corresponding to each diagram. At this point, we check the text similarity between each pair of pervious constructed classes. The system will report the similarity between diagram/classes vocabularies. This report itself represents the different textual inconsistency and incompleteness in our models. By correcting these problems manually if any we able to repeat the last steps again until we obtain a consistent UML models.

Here, we are restricted ourselves to check the similarity of vocabularies not a long text, however, the same manner will be done well if some of the vocabularies of diagrams are expressed as a long text not words.

Returning to our running example, if the system gets the educational system class diagram that is given in Figure 1 and the educational system sequence diagram (B) that is given in Figure 3, the system will report the following inconsistencies:

(i) *Professor* is entailed from *Lecturer.*

(ii) *Division* is entailed from *Department.*

(iii) *Course* is entailed from *Module.*

5 Related Work

There are some works that combine the natural language processing techniques and software development methodology. None of them has been used this new approach mentioned here. For example, Konrad and Cheng [7] identify a round trip engineering process that supports the specification of a UML model using CASE tools, the analysis of specified natural language properties, and the subsequent model refinement to eliminate errors uncovered during the analysis. This process has been implemented in SPIDER, a tool suite that enables developers to specify and analyze a UML model with respect to behavioral properties specified in terms of natural language. This is a configurable process that analysis the UML against specified properties.

Vladimir Mencl [11] describes how readily available tools for natural language

processing can be employed to transform a textual use case into a pro-case in an automated way. There are two premises of this work: (P1) A step of a textual use case describes either communication between an actor and System under design (SuD), or an internal action. (P2). Each action is described by a simple English sentence following a uniform pattern like the SVDPI pattern. This work is mainly concern with the analysis of only one UML diagram, textual UML description diagram, while our work is concerning the consistency amongst the different UML diagrams.

6 Conclusions and Future Works

This article presented the application of the text similarity method in a new application domain, the area of software development. Nowadays, software system is almost modeled using the UML language that becomes the de-facto notation for software engineering projects. However, it is not easy to develop software using UML language from different views without getting inconsistencies or incompleteness problems. To address this problem, we present a framework for checking the consistency among the different UML diagrams. This is done by checking the text semantic similarity among the different vocabularies that are extracted from these various diagrams. The extraction of these vocabularies is done through XMI documents, which are exported automatically from each UML diagram.

Our research to build a strong consistency checker model for software systems through the XMI methodologies is ongoing. In this paper, we simply address the simple vocabularies, like single words. In future work, we intend to use long vocabulary terms that are given by long text or that are described inside the UML comment notes.

References

[1] Bar-Haim, R., Dagan, I., Dolan, B., Ferro, L., Giampiccolo, D., Magnini, B. and Szpektor, I. 2006. The Second PASCAL Recognising Textual Entailment Challenge. In Proceedings of the Second PASCAL Challenges Workshop on Recognising Textual Entailment.

[2] Brown, A. An introduction to Model Driven Architecture. Feb 2004, http://www-128.ibm.com/developerworks/rational/library/3100.html

[3] Corley, C. and Mihalcea, R. Measures of Text Semantic Similarity. In Proceedings of the ACL 2005 workshop on Empirical Modeling of Semantic Equivalence, Ann Arbor, MI, June 2005, pp. 13-18.

[4] Corley, C., Csomai, A. and Mihalcea, R. Text Semantic Similarity, with Applications. In Proceedings of International Conference Recent Advances in Natural Language Processing (RANLP 2005), Borovets, Bulgaria, September 2005.

[5] Dagan, I. Glickman, O. and Magnini, B. 2006. The PASCAL Recognising Textual Entailment Challenge. Lecture Notes in Computer Science, Volume 3944, Jan 2006, Pages 177 - 190.

[6] Engels, G., Kster, J. M. and Groenewegen, L. Consistent Interaction of Software Components. Transactions of the SDPS: Journal of Integrated Design and Process Science, Vol. 6 No. 4. Dec. 2002. pp. 2-22.

[7] Konrad, S. and Cheng, B. H. C. Automated Analysis of Natural Language Properties for UML Models. MoDELS Satellite Events, 2005, LNCS 3844, pp. 48-57.

[8] Kotb, Y. and Katayama, T. Consistency Checking of UML Model Diagrams Using the XML Semantics Approach. 14th International World Wide Web Conference 2005 (WWW 2005), Chiba, Japan, 2005. ACM, pp. 982-983.

[9] Lesk, M. E. Automatic sense disambiguation using machine readable dictionaries: How to tell a pine cone from an ice cream cone. In Proceedings of the SIGDOC Conference 1986, Toronto, June 1986. pp. 24-26.

[10] Lutz, R. R., Targeting Safety-related Errors During software requirements analysis. ACM SIG-SOFT93 Symposium on the Foundation of Software Engineering, 1993, pp. 99-106.

[11] Mencl, V. Deriving Behavior Specifications from Textual Use Cases. In Proceedings of Workshop on Intelligent Technologies for Software Engineering (WITSE04, Sep 21, 2004, part of ASE 2004), pp. 331-341, Sep 2004.

[12] Ossher, H., Tarr, P., Using Multidimensional Separation of Concerns to (Re) Shape evolving Software. Communication of the ACM, October 2001/Vol. 44, No. 10, pp 43-50.

[13] Papineni, K., Roukos, S., Ward, T. and Zhu, W. Bleu: a method for automatic evaluation of machine translation. In Proceedings of the 40th Annual Meeting of the Association for Computational Linguistics (ACL 2002), Philadelphia, PA, July 2002. pp.311-318.

[14] Rocchio, J. Relevance feedback in information retrieval. Prentice Hall, Ing. Englewood Cliffs, New Jersey, 1971.

[15] Salton, G., Singhal, A., Mitra, M. and Buckley, C. Automatic text structuring and summarization. Information Processing and Management, 2(33), 1997. pp. 193-207.

[16] UML Resource Page (UML), Object Management Group (OMG), http://www.omg.org/uml.

[17] Waltz, D. L. and Pollack, J. B. Massively parallel parsing: A strong interactive model of natural language interpretation. Cognitive Science, 9:51-74, 1985.

[18] XML Metadata Interchange (XMI), OMG, http://www.omg.org/technology/documents/formal/xmi.htm