AN INTEGRATED TOOL FOR SUPPORTING ONTOLOGY DRIVEN REQUIREMENTS ELICITATION

Motohiro Kitamura, Ryo Hasegawa
Tokyo Institute of Technology, Ookayama 2-12-1-W8-83, Meguro, Tokyo 152-8552, Japan

Haruhiko Kaiya
Shinshu University, Wakasato 4-17-1, Nagano 380-8553, Japan

Motoshi Saeki
Tokyo Institute of Technology, Ookayama 2-12-1-W8-83, Meguro, Tokyo 152-8552, Japan

Keywords: Requirements Elicitation, Ontology, Requirements Engineering, CASE Tool, Text Mining.

Abstract: Since requirements analysts do not have sufficient knowledge on a problem domain, i.e. domain knowledge, the technique how to make up for lacks of domain knowledge is a key issue. This paper proposes the usage of a domain ontology as domain knowledge during requirements elicitation processes and the technique to create a domain ontology for a certain problem domain by using text-mining techniques.

1 INTRODUCTION

Knowledge on a problem domain where software is operated (simply, domain knowledge) plays an important role on eliciting from customers and users their requirements of high quality. For example, to develop e-commerce systems, the knowledge on marketing business processes, supply chain management, commercial laws, etc. is required as well as knowledge on internet technology. Although requirements analysts have much knowledge of software technology, they may have less domain knowledge. As a result, lack of domain knowledge allows the analysts to produce requirements specification of low quality, e.g. an incomplete requirements specification where mandatory requirements are lacking. Thus, the techniques to provide domain knowledge for the analysts during their requirements elicitation, and computerized tools based on these techniques to support the analysts are necessary.

Kaiya et al. have proposed the methodology called ORE (Ontology driven Requirements Elicitation) (Kaiya and Saeki, 2006) where domain ontologies are used to supplement domain knowledge to requirements analysts during requirement elicitation processes. However, it mentioned just a methodology but did not address the issues on how the analyst can utilize a domain ontology more concretely or on how a domain ontology of high quality can be constructed with less human efforts. This paper presents automated integrated tool for supporting the usage and the construction of a domain ontology. By using this tool, lacking requirements and inconsistent ones are incrementally suggested to the requirements analyst and he evolves a list of the current requirements based on these suggestions. The tool deduces lacking requirements and inconsistency ones by using inference rules on the domain ontology.

Some studies to extract ontological concepts and their relationships by applying text-mining techniques to natural-language specification documents exist (Goldin and Berry, 1997). We apply this technique to various kinds of documents related on a problem domain in order to automate partially the construction of a domain ontology. We customize and adapt usual logical structure of ontologies into requirements elicitation. More concretely, as a result, we adopt varieties of ontological types and their relationships so that an ontology can represent domain knowledge for requirements elicitation. Thus a newly devised text-mining technique fit to our ontological structure is necessary to achieve the construction of domain ontologies from documents.

The rest of this paper is organized as follows. In the next section, we explain the basic idea and show the logical structure of domain ontologies. In section

Kitamura M., Hasegawa R., Kaiya H. and Saeki M. (2007).
AN INTEGRATED TOOL FOR SUPPORTING ONTOLOGY DRIVEN REQUIREMENTS ELICITATION.
In *Proceedings of the Second International Conference on Software and Data Technologies - SE*, pages 73-80
DOI: 10.5220/0001330000730080

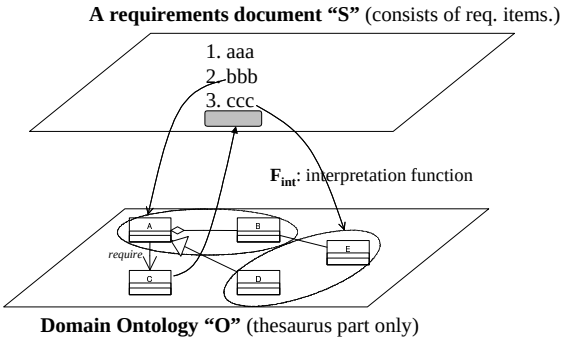


Figure 1: Mapping from Requirements to Ontology.

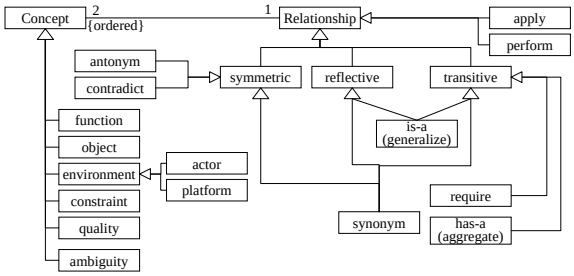


Figure 2: Ontology Meta model.

3, we clarify the tool for supporting our ORE method, i.e. requirements elicitation by using a domain ontology. Section 4 presents the text-mining technique how to create a domain ontology from many documents of this domain, and it also includes an experimental result on how much effort for ontology creation could be reduced by applying our technique. In section 5, we show two case studies to explore the weakness of our tool. In sections 6 and 7, we discuss related works and our current conclusions together with future work, respectively.

2 BASIC IDEA

2.1 Using a Domain Ontology

In this sub section, let's consider how a requirements analyst uses a domain ontology for completing requirements elicitation. Suppose that requirements document initially submitted by a customer are itemized as a list. At first, an analyst should map a requirement item (statement) in a document into atomic concepts of the ontology as shown in Figure 1. In the figure, the ontology is written in the form of class diagrams. For example, the item "bbb" is mapped into the concepts A and B and an aggregation relationship between them. Thus, the ontology plays a role of a semantic domain in denotational semantics. The requirements document may be improved incrementally through the interactions between a requirements analyst and stakeholders. In this process, logical inference on the ontology suggests to the analyst what part he should incrementally improve or evolve. In the figure, although the document S includes the concept A in the item bbb, it does not have the concept C, which is required by A. The inference resulted from "C is required by A" and "A is included" suggests to the analyst that a statement having C should be added to the document S. In our technique, it is important what kind of relationship like "required by" should be included in a domain ontology for inference.

2.2 Domain Ontology

As in (Kaiya and Saeki, 2006), our domain ontology consists of a thesaurus and inference rules. Figure 2 shows the overview of a meta model of the thesaurus part of our ontologies. As shown in the figure, a meta model of thesauruses consists of concepts and relationships among the concepts and it has varies of subclasses of "concept" class and "relationship". For example, "object" is a sub class of a concept class and a relationship "apply" can connect two concepts. Various types of concepts and relationships in Figure 2 are introduced so as to easily represent the semantics in software systems, and it leads to the development of newly devised text-mining technique for creating our ontologies. The detailed explanations of this meta model can be found in (Kaiya and Saeki, 2006).

3 REQUIREMENTS ELICITATION TOOL BASED ON ORE

Based on the technique mentioned in Figure 1, the supporting tool for our ORE method must be able to detect the elements that are incomplete, incorrect, inconsistent or ambiguous, and suggest them to the requirements analysts. To achieve this task, the tool must have powerful reasoning mechanism. We use an SWI Prolog engine because of its flexibility and interoperability to Java. Since in practical environments, requirements are expressed in natural language, we use it to represent the requirements. Like IEEE830 standard (IEEE, 1998) and use case templates, there are some forms or prescribed document structures to write requirements in natural language, however we

don't use them for keeping applicability as wide as possible. Instead, we use hierarchical itemized sentences which are connected with each other in AND-OR style. Itemized sentences are widely used in documents.

In ORE method, the two tasks; 1) developing a mapping between requirements and concepts and 2) analyzing requirements by using a domain ontology should be supported. Figure 3 illustrates a screenshot of the tool for requirements elicitation following the ORE method, and we will use it to explain the elicitation process. The example problem used here is a library system and its initial requirements are displayed in the left upper area of the window. In the right upper area, a thesaurus part of the ontology "Library System" is depicted in class diagram form. Note that our tool is for Japanese and the examples in this paper are the result of direct translation from Japanese into English.

3.1 Mapping between Requirements and Ontological Concepts

To process the meaning of each sentence in requirements symbolically, a set of ontological concepts is related to each sentence in the requirements list. As shown in Figure 3, the initial requirements of the library system consists of three itemized sentences. Lexical and morphological analysis is automatically executed on the sentences so as to extract relevant words from them. We use an automated morphological and syntax analyzer called "Sen" [1] written in Java in order to detect morphemes and to identify their parts of speech (lexical categories such as nouns, verbs, adjectives, etc.). After filtering out insignificant morphemes (e.g. "be"-verb, articles, particles and prepositions) and identifying words and their parts of speech, the tool finds corresponding ontological concepts to each morpheme using a synonym dictionary.

We illustrate the above procedure by using a sentence item # 2 in Figure 3. After morphological analysis, an analyst can get the five words "books", "shall", "highly", "available", and "members" as shown in the first column "word" of the center table of the figure. And then the tool constructs automatically two mappings from the word "books" to the concept "a book" and from "members" to "member", because the concepts having the same labels as these words are included in the ontology shown in the right window. See the columns "word" and "concept1" in the center table "Result" of Figure 3. However, the analyst finds that the word "Books" semantically implies "a copy of a book" in addition to "a book", and he or she tries to modify the mapping from the word "Books" by clicking the class "a copy of a book" in the right window of the figure showing the ontology. He or she clicks the button "set" in the bottom of the right window, and then the new mapping from "Books" to "a copy of a book" is added. Finally he or she can have the mappings 1) "books" → object(a book), object(a copy of a book), 2) "available" → quality(availability) and 3) "members" → actor(member).

3.2 Analyzing Requirements by Using an Ontology

By using concepts and morphemes corresponding to each requirements item, requirements are improved incrementally in this task. Our tool detects 1) the concepts to be added into the current requirements and 2) the concepts to be removed from the current requirements, by using inference on the ontology, and advises suitable actions, e.g. adding a new requirement or removing the existing one, to the requirements analysts. We define the inference rules as Prolog programs [2].

By using the mapping mentioned in the previous sub section, we will intuitively explain how the inference rules can find the lacks of mandatory requirements, i.e. the requirements to be added. See the Figure 3 and the ontology of Library domain shown in Figure 5 (b). In our example, a requirements item #1 is mapped to function(check out). However, no items are mapped to object(member account) even though there is a require-relationship between function(check out) and object(member account) as shown in Figure 5 (b). Our tool detects the missing of "member account" by the inference on this require-relationship, and gives an advice to add object(member account) automatically. Following this advice, an analyst adds new requirements item about member account, e.g., "Each member has his/her own member account". The rule A1 shown in Figure 4 is written based on this idea. In the figure, we illustrate a part of these rules for producing advices to requirements analysts, and they can be recursively applied to produce the advices. The rules are represented with Horn clauses of Prolog. They can be categorized with respect to their contribution to characteristics of a good Software Requirements Specification of IEEE 830 (IEEE, 1998), and for example, rules A1, A2, A3, A4 and A5 for completeness, and A6 for unambiguity.

Let's turn back to our example, Figure 3. When

[1] https://sen.dev.java.net/

[2] http://www.swi-prolog.org

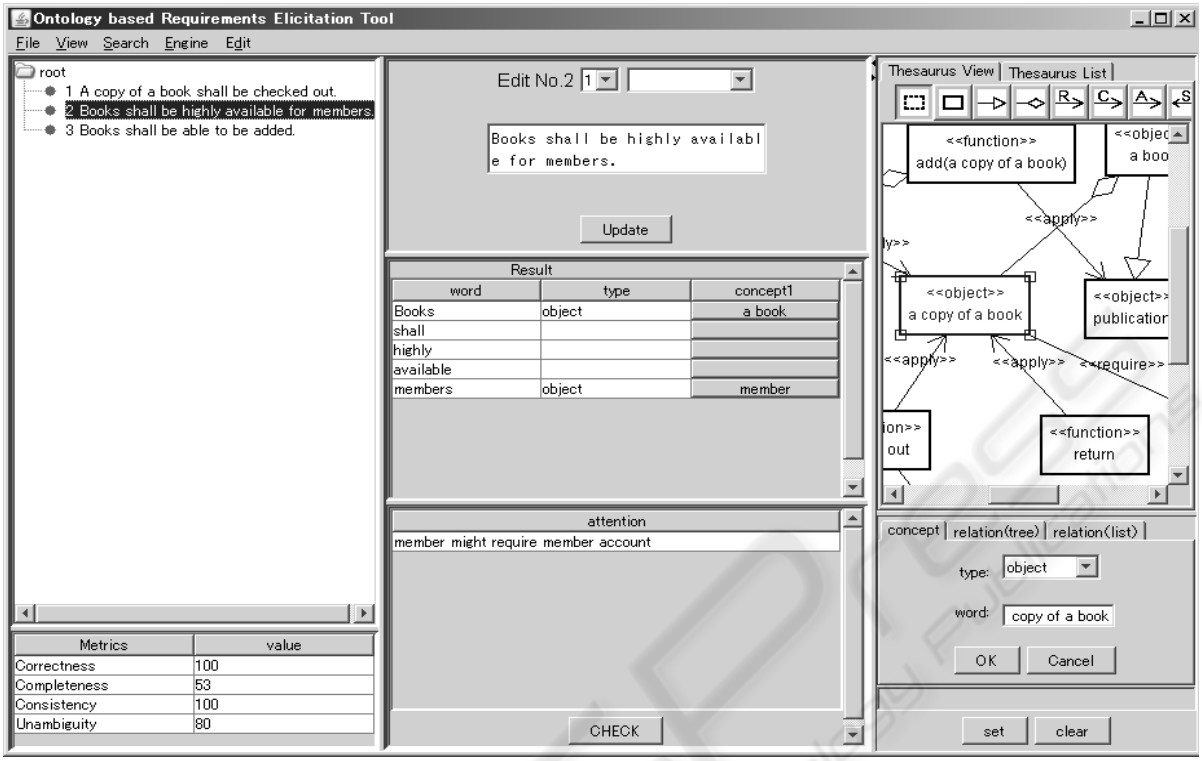


Figure 3: A Tool for Requirements Elicitation.

Rule A1: If concepts(all) includes concept(A), concepts(all) does not include concept(B) and Ontology() includes require(concept(A), concept(B)), then our tool gives an advice to add or modify requirements item(s) so that concepts(all) includes concept(B).

Rule A2: If concepts(all) includes object(A), concepts(all) does not include function(B), and Ontology() includes apply(function(B), object(A)), then our tool gives an advice to add or modify requirements item(s) so that concepts(all) includes function(B).

Rule A3: If concepts(all) does not include object(A), concepts(all) includes function(B), and Ontology() includes apply(function(B), object(A)), then our tool gives an advice to add or modify requirements item(s) so that concepts(all) includes object(A).

Rule A4: If concepts(all) includes environment(A), concepts(all) does not include function(B), and Ontology() includes perform(environment(A), function(B)), then our tool gives an advice to add or modify requirements item(s) so that concepts(all) includes function(B).

Rule A5: If concepts(all) does not include environment(A), concepts(all) includes function(B), and Ontology() includes perform(environment(A), function(B)), then our tool gives an advice to add or modify requirements item(s) so that concepts(all) includes environment(A).

Rule A6: If morpheme x is identical to morpheme y but mapc(morpheme x) is not identical to mapc(morpheme y), then our tool gives an advice to rename one of these morphemes.

Abbreviations: concepts(item x): a set of concepts related to a requirements item x. concepts(all): the union of the sets of concepts related to all requirements items in a requirements list. ($\cup_x$ concepts(item x)). morphemes(item x): a set of morphemes related to an item x. mapc(morpheme x): a set of concepts such that the mapping between x and the concept exists. Ontology(): a set of all relationships and concepts of a domain ontology.

Figure 4: Inference Rules for Producing Advices.

the analyst clicks a CHECK button located in the bottom of the center area in the figure, the tool starts inference to detect lacking requirements, inconsistent ones and ambiguous ones. In this example, according to the ontology "member" requires the concept "member account" to distinguish members of the library from non-members, and the rule A1 mentioned above suggests that "member account" should be added. This inference result is shown in the bottom area "attention" in the figure, and he or she can add the sentence related to "member account" at the top area "Edit".

Our tool also tells us the measures of correctness, consistency, unambiguity and completeness with the ratio of the detected faults to total number of requirements items. Their calculation technique can be found in (Kaiya and Saeki, 2006). The left bottom window of Figure 3 shows the calculation results of these values. They help an analyst to decide when she may terminate her elicitation task.

4 A SUPPORTING TOOL FOR ONTOLOGY CREATION

In this section, we focus on the technique to extract a thesaurus from Japanese text documents. Basically, nouns and verbs included in the documents correspond to the object concepts and functions of Figure 2 respectively, and adjectives and adverbs modifying objects or functions represent the concepts of quality. Thus the essential parts of our process for ontology creation are 1) Word extraction for extracting from the documents the significant words and terms that can be considered as ontological concepts, and 2) Relationship extraction for discovering the relationships among the extracted words and terms (we simply call them words). After morphological analysis and dependency analysis, we identify part-of-speech categories of the meaningful words appearing the documents such as nouns, verbs, adjectives etc. These steps can be performed automatically using the natural-language processing tool Sen, which is used in section 3. By using part-of-speech information of words, we classify the words into the types of ontological concepts such as object, function and quality. For example, "Customer Information" is a noun and is classified into an object concept.

In the next step, i.e. Word extraction, our tool calculates various measure parameters of the words so that we can filter out unimportant words from the classified words. The parameters that we use are based on word frequency, i.e. the number of times a word appears in documents, and are shown below.

1. TF (term frequency): the number of times a word appears in the documents.
2. TF $\times$ IDF (term frequency $\times$ inverse document frequency) : the term frequency of a word weighted with its importance degree resulting from the number of the documents it appears.
3. Entropy: logarithmic value of the term frequency of a word weighted with its entropy.
4. C-value : the term frequency of a word weighted with its length and its occurrences as a part of multi-words.

Figure 5 (a) shows an example of the result of word extraction. As shown in the figure, the words are measured and they are sorted in descending order of the measure values. An ontology creator can select the important words denoting ontological concepts in a problem domain, by checking boxes on the sorted list of the measured words. In the example of the figure, the words "publish", "lending", "search", "print" and "return" have been manually selected by the creator.

After selecting the words, the creator proceeds to the step of extracting relationships among the words. The supporting tool calculates the number of times a pair of words included in a sentence appears in the documents, i.e. co-occurrence frequency (CF) of two words and cosine similarity (CS) of co-occurrence frequency vectors, in order to find semantically relevant word pairs. After calculating CFs and CSs, pairs of words whose CF and CS are higher than certain thresholds are selected as candidates of ontological relationships. Based on types of words (e.g. object, function and quality) and dependency structures in the sentences, the tool suggests the types of the ontological relationships. For example, suppose that the CF of the word u and v is higher, and u and v are "object" and "function" respectively. In addition, if u is an object in grammatical sense and v is its verb, the tool suggests an "apply" relationship between u and v. Cosine similarity is useful to detect synonyms, is-a and has-a relationships. Figure 5 (b) shows an example of the detected concepts and their relationship of Library domain in a class diagram-like form. The tool users can modify the detected relationships and edit the diagram to make it more complete and precise as a domain ontology.

To assess our thesaurus-creation tool, we made an experiment to measure the effort of worker's activities. We had 8 documents on software for making estimates, whose lengths were from 3 to 23 pages of A4 paper size and used them as inputs of the tool to create a thesaurus of "making an estimate" domain. We had three subjects, and one of them developed a domain thesaurus manually by referring to the 8 doc-

Table 1: Comparative Results Between S1 and S2 in Case Study 1.

	Initial	S1	S2
Number of requirements items	14 (1.0)	60 (4.2)	44 (3.1)
Number of mapped concepts	19 (1.0)	39 (2.0)	34 (1.7)

uments. He completed the thesaurus after three days, and might spent totally more than 24 hours. Two of our subjects used the tools and they spent 260 and 180 minutes respectively in creating their domain thesauruses. They created 432 and 363 relationships between their extracted words respectively, and 147 of them were the same. These results can lead us to the conclusion that our tool significantly helps in efficient thesaurus-creation, although manual activities still remain.

5 CASE STUDIES

In this section, we present experimental results obtained from two case studies of requirements elicitation in order to assess our tool.

5.1 Case Study 1

In our first case study, two skilled students elicited requirements for "a record management system in a school". They did not know the details of this system, but were its users. One student S1 used our tool to elicit requirements, while another student S2 did not. By comparing their results and processes, we discuss the advantages and disadvantages of our tool for requirements elicitation.

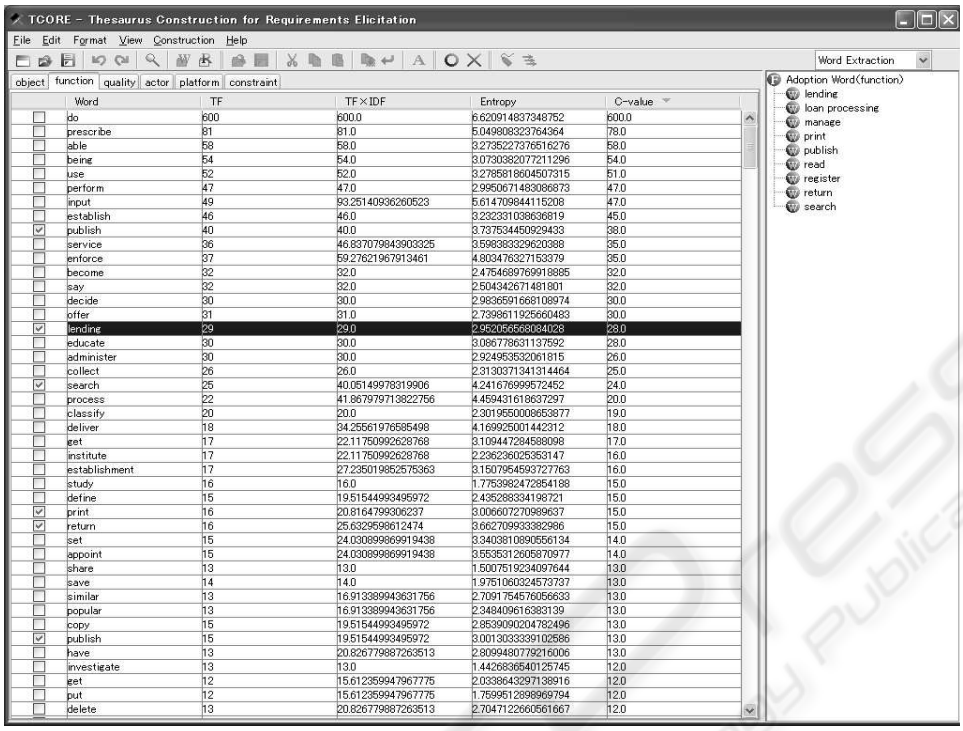
The comparative result is shown in Table 1. The initial list of requirements that was given to our subjects consisted of 14 itemized simple sentences. Through the requirements elicitation processes, S1 and S2 got 60 and 44 itemized simple sentences as final requirement lists respectively. The second row in the table expresses how many concepts the final requirements items were mapped to. Although S2 did not create the mappings from the sentences to the concepts of the thesaurus during his requirements elicitation process, we asked him to make the mapping after this experiment so as to get this second row of Table 1. For example, the 60 itemized sentences that S1 produced were mapped into 39 concepts of the thesaurus during the experiment, while S2 mapped the 44 sentences into 34 concepts after the experiment was finished. Since each item in the requirements list can have different amount of its meaning, we calculated and compared the number of concepts related to each requirements list. Each number in parentheses in the table shows the ratio to initial items. For example, the number of requirements item was extended about 4.2 times of the number of initial requirements items. The table 1 suggests that our tool can significantly help an analyst to extend the requirements list, in particular to find the lacks of requirements, because the number of items was extended about 4.2 times by S1 but 3.1 times by S2. With respect to the amount of mapped concepts, the difference is not so significant: 2.0 and 1.7. It can be considered that many items that S1 produced referred to the same concepts but they were described in detail and were further refined than S2.

5.2 Case Study 2

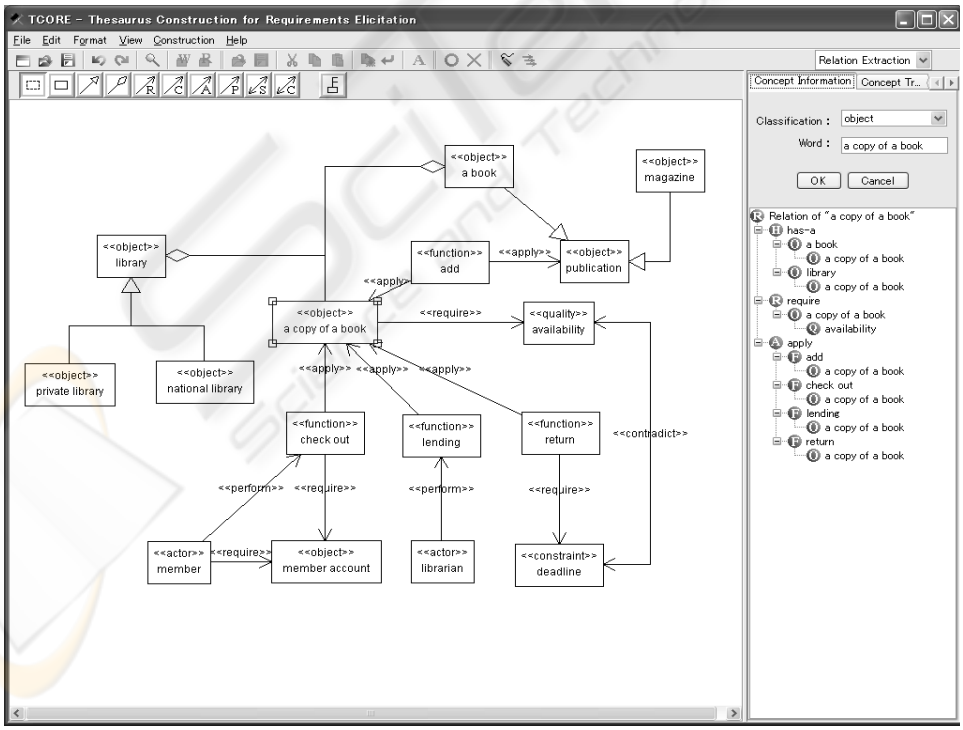
The second case study used a subject unfamiliar to a problem domain in order to assess our ORE. We set an example of a POS (Point Of Sales) system for convenience stores. The subject was a skilled student for software design and programming but did not have knowledge on business processes of convenience stores or on POS systems. He was given 6 initial requirements and an ontology of POS domain having 43 concepts and 49 relationships, which had been created with our tool. Finally he elicited 31 requirements items (incl. the initial 6 items) after 2 hours. 7 items of newly added 25 (31-6) items resulted from the thesaurus part of the ontology (which was indicated in the right area of the tool screen as shown in Figure 3), and 4 items from the advices that our tool suggested. This result that 44% items were suggested with the ontology showed the useful contribution to requirements elicitation by domain non-experts. Since the size of the ontology was small, the subject did not pay much attention to the advices derived from the inference, but rather to the list of the words indicated in the right area of the tool window so as to get a full view of the significant domain-specific words.

6 RELATED WORK

In research community of Ontology, many computerized tools for supporting ontology creation have been developed, and almost of them were diagram editors with simple syntactic checking mechanisms such as KAON and Protégé where users can input and edit ontologies visually. Text2Onto of KAON (Cimiano and Volker, 2005) is a pioneer of the tools having a text-mining function based on TF $\times$ IDF measure so that words frequently appearing can be extracted from text documents. In fact, our ontology creation tool



(a) Word Extraction



(b) Relationship Extraction

Figure 5: Ontology Creation Tool.

uses the same quantification techniques for word extraction. However, our ontologies have varieties of types of concepts and of relationships in order to apply to requirements elicitation, and Text2Onto cannot classify the extracted concepts and relationships into these types. OntoLearn (Navigli et al., 2003) uses WordNet to detect semantic relationships among extracted words and this technique can be applied in order to make our tool more elaborate. Activity First Method (AFM) (Mizoguchi et al., 1995) is a methodology to extract task ontologies from natural language texts manually. Unlike ours and Text2Onto, it adopted the approach based on the occurrences of verbs in the texts so as to construct the conceptual structures of tasks corresponding to the verbs. In (Kof, 2004), a case study to try to extract an ontology from a requirements document was presented, and the extracted ontology has been used as a formal design model of a software system to be developed. Its aim of this work is different from ours because a domain ontology is not considered as domain knowledge in it.

We could find several studies on the application of ontologies to requirements engineering. LEL (Language Extended Lexicon) (Breitman and Leite, 2003) is a kind of electronic version of dictionary that can be used as domain knowledge in requirements elicitation processes. Although it includes tags and anchors to help analysts fill up domain knowledge, it has neither methodologies as guidance procedures nor semantic inference mechanisms. A feature diagram in Feature Oriented Domain Analysis can be considered as a kind of a domain thesaurus representation, and in (Zhang et al., 2005), a technique to analyze semantic dependencies among requirements by using features and their dependency relationships was proposed. However, its aim is different from ours.

7 CONCLUSION

The tool that we developed includes two contributions; the first one is that the inference mechanism implemented with Prolog helps a requirements analyst to evolve requirements systematically by taking account of the semantic aspect of requirements, and the second one is that the tool supports domain ontology creation by using integrated metrics for text-mining. We partially assess the user-friendliness and effectiveness of our tool through experiments. However, our experiment mentioned in section 5 was too small to argue the generality of the experimental findings.

The quality of requirements elicitation using our tool greatly depends on the quality of domain ontologies. Although we adopted text-mining approach from the existing documents such as manuals, we have to improve the approach moreover. Although our current approach is based on the frequency of words in documents, frequent words are not always important in general. Comparing different documents (Lecceuche, 2000) is one of the ways to complement the frequency based approach. Another way to create an ontology of higher quality is the integration of many ontologies existing over Internet, which have been developed by XML, OWL and Ontology community.

REFERENCES

Breitman, K. and Leite, J. (2003). Ontology as a Requirements Engineering Product. In *Proc. of 11th IEEE Requirements Engineering Conference (RE01)*, pages 309–319.

Cimiano, P. and Volker, J. (2005). Text2onto : A framework for ontology learning and data-driven change discovery. In *Lecture Notes in Computer Science*, volume 3513, pages 227–238.

Goldin, L. and Berry, D. (1997). AbstFinder, A Prototype Natural Language Text Abstraction Finder for Use in Requirements Elicitation. *Automated Software Engineering Journal*, 4(4):375 – 412.

IEEE (1998). IEEE Recommended Practice for Software Requirements Specifications. IEEE Std. 830-1998.

Kaiya, H. and Saeki, M. (2006). Using domain ontology as domain knowledge for requirements elicitation. In *Proc. of 14th IEEE International Requirements Engineering Conference (RE'06)*, pages 189–198.

Kof, L. (2004). Natural Language Processing for Requirements Engineering: Applicability to Large Requirements Documents. In *Proc. of the Workshops, 19th International Conference on Automated Software Engineering*.

Lecceuche, R. (2000). Finding Comparatively Important Concepts between Texts. In *The Fifteenth IEEE International Conference on Automated Software Engineering (ASE'00)*, pages 55–60, Grenoble, France.

Mizoguchi, R., Ikeda, M., Seta, K., and Vanwelkenhuysen, J. (1995). Ontology for modeling the world from problem solving perspectives. In *IJCAI Workshop on Basic Ontological Issues in Knowledge Sharing*.

Navigli, R., Velardi, P., and Gangemi, A. (2003). Ontology learning and its application to automated terminology translation. *IEEE Intelligent Systems*, 18(1):22–31.

Zhang, W., Mei, H., and Zhao, H. (2005). A Feature-Oriented Approach to Modeling Requirements Dependencies. In *Proc. of 13th IEEE International Conference on Requirements Engineering (RE'05)*, pages 273–284.