Using Domain Ontology as Domain Knowledge for Requirements Elicitation

Haruhiko Kaiya
Shinshu University, JAPAN
kaiya@cs.shinshu-u.ac.jp

Motoshi Saeki
Tokyo Institute of Technology, JAPAN
saeki@se.cs.titech.ac.jp

Abstract

Domain knowledge is one of crucial factors to get a great success in requirements elicitation of high quality, and only domain experts, not requirements analysts, have it. We propose a new requirements elicitation method ORE (Ontology based Requirements Elicitation), where a domain ontology can be used as domain knowledge. In our method, a domain ontology plays a role on semantic domain which gives meanings to requirements statements by using a semantic function. By using inference rules on the ontology and a quality metrics on the semantic function, an analyst can be navigated which requirements should be added for improving completeness of the current version of the requirements and/or which requirements should be deleted from the current version for keeping consistency. We define this process as a method and evaluate it by an experimental case study of software music players.

1 Introduction

Domain knowledge that experts in a problem domain have plays an important role on eliciting requirements of high quality. Although requirements analysts having much knowledge of software technology, they don't know or understand their problem domain where software to be developed will be operated. Lack of domain knowledge allows the analysts to perform poor requirements elicitation and as a result, they come to produce requirements specification of low quality. Requirements elicitation from stakeholders is actually one of the most crucial steps in requirements analysis processes, and several methods and computerized tools have been studied and developed in order to support human activities of requirement elicitation, e.g., goal oriented requirements analysis methods, scenario analysis, use case modeling techniques and so on. However, these methods and tools are too general, in the sense that they are for the general problem domains where problem-specific domain knowledge is not used and that they did not support the utilization of domain knowledge. To achieve more efficient supports for requirements elicitation of higher quality, we need a method and tools incorporating with the support of utilizing domain knowledge. In this research direction, one of the major issues is the technique to model and to represent domain knowledge for requirements elicitation. Our approach uses a *domain ontology* as a representation of domain knowledge.

Ontology technologies are frequently applied to many problem domains nowadays [8]. Because concepts, relationships and their categorizations in a real world can be represented with ontologies, they can be used as resources of domain knowledge. As mentioned in [13], we consider an ontology as a thesaurus of words and inference rules on it, where the words in the thesaurus represent concepts and the inference rules operate on the relationships on the words. The two major benefits of applying ontology as domain knowledge can be considered as follows.

The first benefit is related to semantic processing of requirements descriptions by computer. To support requirements elicitation effectively by computerized tools, semantic processing on requirements descriptions is necessary. Especially, it is significant that we can detect incompleteness and inconsistency in requirements to an information system to be developed. Usually both of an initial requirements list, which a customer brings up to a development organization at first, and a requirements specification document as a final artifact are written in natural language, i.e., are informal descriptions. Although the techniques for natural language processing (NLP) are being advanced nowadays, it is hard to handle with this informality such requirements documents sufficiently by computers. There are several approaches to tackle this problem. Some studies included a topic on a semi-formal notation for representing requirements, e.g., restricted natural languages, so that automated semantic processing is possible, but it was difficult for human engineers to write syntactically and semantically correct requirements sufficiently by using this notation. Rigorous formal notations with axioms and inference system seem to be suitable for automate semantic processing, but its usage is very limited to practitioners because of their difficulty and complexity in the practitioners' learning and training. A domain ontology allows us to have a semantic basis for requirements descriptions and to achieve "lightweight semantic processing" in order to detect properties of requirements descriptions. Each concept of the domain ontology can be considered as a semantic atomic el-

14th IEEE International Requirements Engineering Conference (RE'06)
0-7695-2555-5/06 $20.00

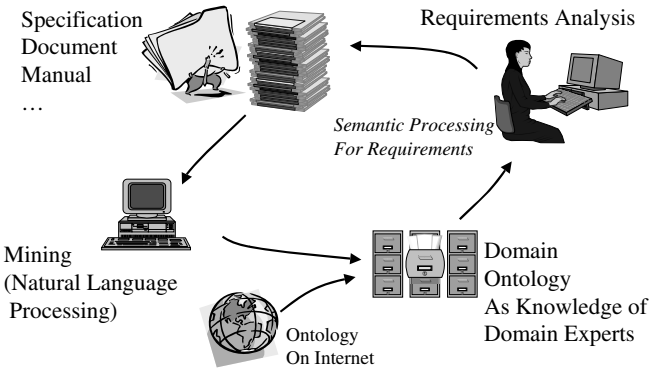

Figure 1. Requirements and Knowledge Elicitation Cycle

ement that anyone can have the unique meaning in a problem domain. That is to say, the thesaurus part of the ontology plays a role of a semantic domain in denotational semantics, and inference rules can detect properties such as inconsistency and incompleteness of requirements descriptions. By using such an ontology, several kinds of semantic processing can be achieved in requirements analysis without rigorous NLP techniques.

Secondly, although we have not a complete ontology for our problem domain yet, we can have several (semi-)automated techniques to do it. Like DAML Ontology Library [1], researchers and developers in knowledge engineering communities, in particular Semantic Web community are constructing many ontologies in wide varieties of domains, and they are represented with standardized OWL based language so as to be exchanged by many persons. Furthermore, some studies to extract ontological concepts and their relationships by applying text-mining techniques to natural-language documents exist [6]. Existing requirements specification documents can also be significant sources to text-mining processing. It means that we can incrementally make a domain ontology and requirements descriptions in a problem domain more complete, as shown in Figure 1. In the figure, a requirements analyst elicits more complete and more consistent requirements by using a domain ontology and documents them as a requirement specification. This document is mined by suing NLP techniques to extract the concepts and the relationships that can be incorporated into the current version of the domain ontology. Through enacting this cycle, we can get both more complete domain ontology and a requirements document of high quality.

In this paper, we focus on the first benefit, i.e. using a domain ontology for semantic processing of requirements descriptions written in natural language. In [18, 9], we have discussed the application of an ontology to semantic processing of requirements descriptions written natural language, more concretely detecting incompleteness and inconsistency. Its main idea is as follows. A requirements analyst explores a mapping between a requirements description and ontological elements such as concepts and their relationships included in a domain ontology. After establishing the mapping, by using inference rules on the ontology system, various kind of semantic processing such as detecting incompleteness and inconsistency and measuring the quality of the requirements descriptions. Suppose that the following example. The analyst can map a requirement "a system can reserve seats in a specified train" to an ontological element "reserve" in our domain system of the domain of reservation systems. Another element "cancel" has a relationship "requires" to "reserve" in our ontology system. Thus our inference system can suggest to the analyst that it was incomplete unless he did not include "cancel" function in his system. This kind of inference on an ontology system is a "lightweight" processing for computers.

This paper presents a technique to make our "lightweight" approach concrete from a methodological viewpoint for requirements elicitation processes. That is to say, we define a procedure as a methodology for requirements analysts to elicit requirements from an initial requirements list written in the form of itemized natural language sentences. The rest of this paper is organized as follows. In the next section, we explain the basic idea how to use a domain ontology and discuss the requirements to an elicitation method based on our approach. The method is called ORE (Ontology based Requirements Elicitation method). In section 3, we clarify our ontology system in detail. We also clarify the procedure how to progress the method, especially how to update the logical structure during the method in section 4. Section 5 presents the assessment of our method by using experimental case studies. In section 6 and 7, we discuss related works and our current conclusions together wit future work, respectively.

2 Using a Domain Ontology for Eliciting Requirements

2.1 What is a Domain Ontology System?

According to [8], the most popular definitions of "ontology" are "formal explicit specification of shared conceptualization" and "classifications of the existing concepts". In this paper, a domain ontology provides semantic basis on requirements to be elicited. More concretely, each statement representing a requirement (we call it requirements statement) can be interpreted with ontological elements and such an ontology should include atomic concepts that any stakeholders can commonly have in a problem domain.

By using lexical decomposition technique [19], each requirements statement can be decomposed into several terms that are interpreted in the same way by anyone. Semantic structure among such terms, i.e., a thesaurus is required for

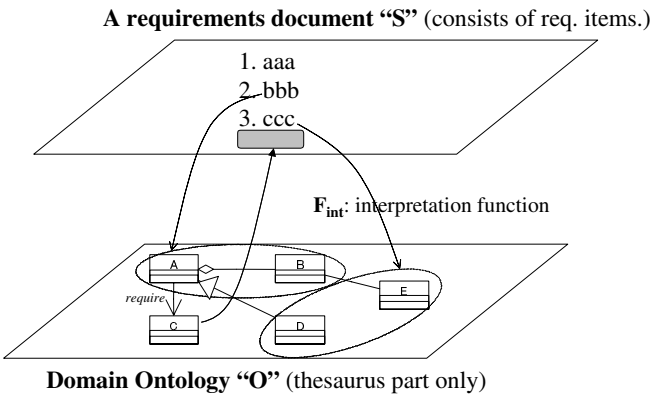

Figure 2. Mapping from Requirements to Ontology

our requirements elicitation because such terms can be used as pointers to concepts. Thus an ontology should include a thesaurus in a problem domain, where atomic terms are connected to each other through specific relations or associations. For example, the atomic concept "reserve" has "requires" relationship to "cancel" in the domain of Reservation Systems. To detect incompleteness and inconsistency, it should be possible to infer the properties of requirements by using these relationships. Suppose that elicited requirements include a statement interpreted as the concept "reserve" only and do not have "cancel". In this case, the inference using "requires" relationship suggests that the requirement related to "cancel" may be missing. This kind of inference and atomic concepts represented with a thesaurus depends on a problem domain. Thus we can consider that a domain ontology consists of a domain specific thesaurus and inference rules on it. The details of an ontology and inference rules will be mentioned in section 3.

2.2 How is an Ontology Used?

As mentioned in section 1, an ontology plays a role of a semantic domain in denotational semantics.

Below, let's consider how a requirements analyst uses a domain ontology for completing requirements elicitation. Suppose that a requirements document initially submitted by a customer are itemized as a list. At first, an analyst should map a requirement item (statement) in a requirement document into atomic concepts of the ontology as shown in Figure 2. 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. 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 describe. In the figure, although the document S includes the concept A at 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.

By using our approach, we can estimate the quality of requirements. We pick up four quality characteristics as follows and summarize intuitive definitions. For the readers having an interest to their formal definitions, see [9].

1. Correctness (COR): The requirements items that were mapped into ontological elements can be considered as requirements appropriate for a problem domain.

$$\mathrm{COR} = \frac{\#\{\text{requirements items that are mapped into the ontology}\}}{\#\{\text{requirements items (total number of requirements items)}\}}$$

2. Completeness (CMP) : The ontological elements that did not have any mapped requirements items can be candidates for missing requirements items.

$$\mathrm{CMP} = \frac{\#\{\text{ontological elements that no requirements items are mapped into}\}}{\#\{\text{ontological elements (total number of ontological elements)}\}}$$

3. Consistency (CST) : If requirements items include ontological concepts that are connected through "contradict" relationship in the ontology, they are inconsistent.

$$\mathrm{CST} = \frac{\#\{\text{ontological relationships that are not "contradict" and that some requirements items are mapped into}\}}{\#\{\text{ontological relationships that some requirements items are mapped into}\}}$$

4. Unambiguity (UAM): If a requirements item is mapped into several concepts that have no semantical relationships, it can have multiple meaning.

$$\mathrm{UAM} = \frac{\#\{\text{the requirements items that are mapped into concepts that can be traced from each other through relationships}\}}{\#\{\text{requirements items}\}}$$

"Semantical relationship" between concepts means the reachability of the concepts by using relationships on the ontology.

We use the values of these measurements in ORE method for navigating analyst's activities.

3 Ontology Systems and Inference Rules

To elicit and define requirements systematically, we have to formally define the logical structure of artifacts that are

used in requirements elicitation processes, i.e., requirements lists, ontologies and semantic mappings from a requirement list to an ontology in this section.

3.1 Requirements List

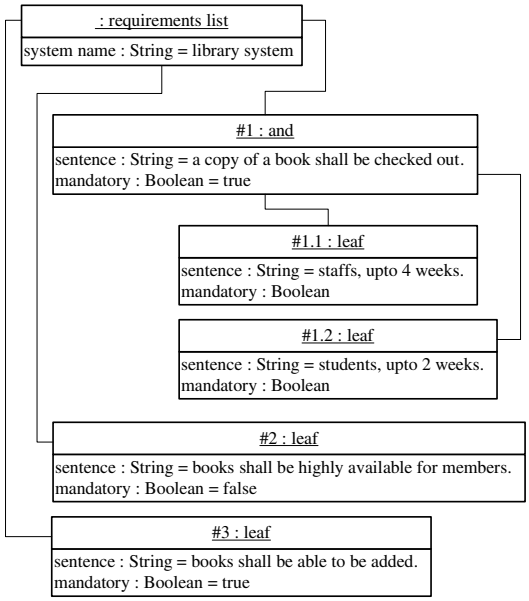

Figure 3. A Requirements List

The input of our requirement elicitation process is a list of the natural-language sentences, called requirements list. A requirements list consists of itemized sentences written in natural language, and each sentence represents a customers' requirement to a software system to be developed. A requirements list consists of hierarchically itemized sentences and we call them "requirements items" or simply items. The itemized sentences in the same level of hierarchy can be semantically connected with either "and" and "or" relationship. Figure 3 shows an instance of a requirements list for a part of a library system. For example, the sub-graph whose root node is #1 denote the requirements list below;

1. a copy of book shall be checked out.
   1.1. staffs, up to 4 weeks and
   1.2. students, up to 2 weeks.

3.2 Representing an Ontology

To manipulate requirements items semantically, we have to map each requirements item into semantic elements, which is a similar way to denotational semantics. These semantic elements are defined on an ontology in our research. Figure 4 shows the overview of a meta model of the thesaurus part of our ontologies [9]. As shown in the figure, thesauruses consist of concepts and relationships among the

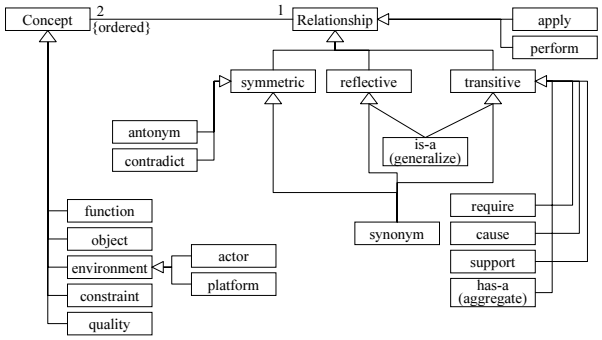

Figure 4. Ontology Meta model (Thesaurus Part)

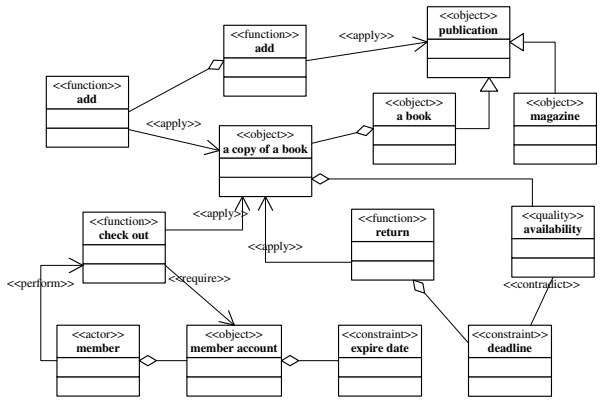

Figure 5. A Part of a Domain Ontology for Libraries

concepts and they have varies of subclasses of "concept" class and "relationship". In the figure, "object" is a sub class of a concept class and a relationship "apply" can connect two concepts. Concepts and relationships in Figure 4 are introduced so as to easily represent the semantics in software systems. Intuitively speaking, the concepts "object", "function", "environment" and their subclasses are used to represent functional requirements. On the other hand, the concepts "constraint" and "quality" are used to represent non-functional requirements. The concept "constraint" is useful to represent numerical ranges, e.g., speed, distance, time expiration, weight and so on.

Basically, an instance of our ontology is represented in a directed typed graph where a node and an arc represent a concept and a relationship (precisely, an instance of a relationship) between two concepts, respectively. Figure 5 shows a part of an instance of an ontology of library information systems, and we use class diagram notation. Stereo types attached to boxes (nodes) and arrows (arcs) show their types. For example, "check out" belongs to a "function" concept of Figure 4. An arc between "check out" and "a

copy of a book" is an "apply" relationship, which presents that an object "a copy of book" participates in the function "check out".

As will be mentioned later, we use Prolog to infer various properties on an ontology and so we should represent the instance as Prolog fact. Actually, the example ontology of Figure 5 can represented with predicates as follows;

```
apply(function(check out), object(a copy of a book))
apply(function(return), object(a copy of a book))
apply(function(add), object(a copy of a book))
apply(function(add), object(publication))
has-a(object(a copy of a book), quality(availability))
has-a(function(return), constraint(deadline))
has-a(object(member account), constraint(expire date))
...
```

The names of predicates come from relationships. For example, in "has-a" relationship, its first parameter stands for a "whole" concept, while the second is a "part" of the "whole" object, e.g. the predicate has-a(object(a copy of a book)) expresses that a copy of a book has the quality characteristic "availability". To clarify the type information on concepts explicitly in Prolog facts, we use unary functions whose names are the types of the concepts, e.g. function(check out) expresses that the type of the concept "check out" is "function". The descriptions related to qualities and constraints sometimes contradict or contribute to each other, and we have two relationships "contradict" and "contribute" in our ontologies. For example, in Figure 5, the following relationship can be found.

contradict(constraint(deadline), quality(availability))

It means that extending the deadline of returning borrowed books causes less availability of them to other potential borrowers. Some relationships are categorized into reflective, transitive or symmetric types in these figures. We can infer several facts based on such categorization. For example, the relationship "contradict" belongs to symmetric types, so we can have the following inference rule automatically:

contradict(x,y) $\rightarrow$ contradict(y,x)

3.3 Mapping and Inference Rules

To utilize the ontology mentioned in the previous sub section, we have to discuss how to represent the semantic mappings that were shown in as shown in Figure 2. Figure 6 illustrates semantic mappings from the requirements items to a Library ontology, and the instances of "map item" specify semantic mappings from a requirements item to ontological elements, i.e. concepts and relationships. For example, the item "1. a copy of a book shall be checked out" is mapped to the ontological elements "a copy of a book" and "check out" through the map item #1. And we can consider that the requirements item # is also mapped to "apply" relationship between "a copy of book" and "check out". The concepts and the relationships that a requirements item is mapped to is called *mapped concepts* and *mapped relationships* respectively. Furthermore, we call together both of mapped concepts and mapped relationships *mapped elements*. In this example, the mapped concepts of a requirements item #1 are both function(check out) and object(a copy of a book). Although the automated techniques to find such mappings is out of scope of this paper, we can consider that lexical matching and keyword matching using thesauruses in the area of information retrieval are one of the promising techniques. We can calculate measures mentioned in 2.2 using mappings of Figure 6.

Based on mapped elements, we clarify the meaning of requirements items on an ontology. By using the inference rules in the ontology, we extend and improve the requirements list. Our inference rules are written in simple if-then form such as "if require(X,Y) exists in the ontology and X is a mapped element, there is a requirements item that is mapped to Y". In Figure 6, we have the relationship "require(function(check out), object(member account))" and the mapped concept "check out" to which the "#1 leaf" is mapped. Following the above rule, we should include "object(member account)" as a mapped concept, and as a result extend the requirements list by adding the sentence referring to "member account". Another example of the rules is "if contradict(X,Y) exists in the ontology and X and Y are mapped elements, then some requirements items that are mapped to X or to Y should be deleted so as to include either X or Y". This deletion allows us to have the consistent requirements list. We will explain such procedure in detail in the next section.

Note that our semantic mapping approach does not provide precise meaning of natural language sentences, but that it clarifies *references* to a fragment of atomic meaning concepts. Consider the example of the following two requirements items; the deadline is 4 weeks later and the deadline is 2 weeks later. If our ontology includes the concepts of the numbers "4" and "2", we can distinguish between them. However, the ontology shown in Figure 5 has the concept "deadline" only, and thus both of these items are mapped into it. As a result, we cannot specify their semantic difference on the ontology and this semantic mapping. The mapping provides *references* of a requirement to a part of meaning.

4 Procedure in Our Method

In this section, we clarify a procedure of our method ORE. A requirements analyst follows it to produce a complete set of requirements items from an initial requirements list. The initial requirements list is provided by customers and/or users, and it may include inconsistent and ambiguous descriptions. Furthermore greatly indispensable requirements items may be missing in the initial list, and the items that should be further refined and made more concrete may still remain in the list. A requirement analyst uses our method to extend and improve a requirements list

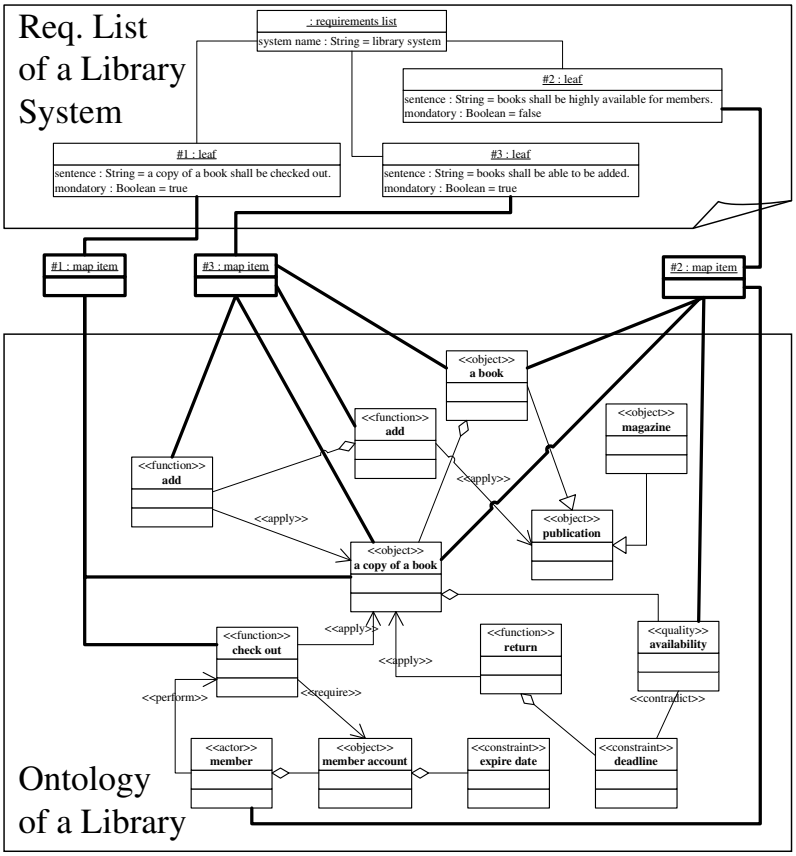

Figure 6. A Part of a Semantic Mapping

incrementally and iteratively, interacting with stakeholders such as customers and users by interviews.

Figure 7 shows the process view of our method to improve and extend requirements. After making mappings in Step 2 in Figure 7, the metrics mentioned in section 2 can be calculated automatically, but accepting the calculating results depends on the decision of a requirements analyst. For example, even if the metrics values are not high enough, an analyst does not have to update requirements. The metrics and evaluation results are merely suggestions or hints to update the requirements list. When the size of requirements list and/or ontology increases, the cost of automatic calculation may also increase. However, we do not think that it causes to increasing the amount of analyst's tasks, and we need more experiments and/or simulations to clarify this point.

Basic idea of our method, especially Step 4 in the figure, is as follows. At first, we estimate the quality of a requirements list by using the technique mentioned in section 2.2. Based on the estimation results, we select alternatives to improve and to extend the list. For example, when the completeness value of the list is not high, we find what items are missing and should add them. An ontology helps requirements analysts to find these items as mentioned in 2.2. On the ontology, we trace the relationships such as "require" and "apply" associated with the concepts corresponding to the items so that we can find the concepts to be newly added. Step 4.(a) in Figure 7 has been developed based on this idea.

Consider that two sets A and B of concepts where no concepts in A can be traced from a concept in B through relationships "require" or "apply" and vice versa on the ontology, so called "islands" isolated to each other. The concepts included in an island have no semantically relationships to the concepts of another island. If a requirements item is mapped into two different islands, the item can be interpreted to have two meanings. Our method helps a requirements analyst to find the items mapped into more than one island. The analyst can decrease the ambiguity of the item by removing such mappings into more than one island. Step 4.(b) in Figure 7 is designed based on these ideas.

Our inference rules on a thesaurus written in Prolog calculate four measures as well as deduce the candidates for missing requirements items, ambiguous, inconsistent and incorrect ones, by tracing "is-a", "has-a", "apply", "require" and "contradict" relationships and by extracting "islands".

After completing the requirements list, requirements

1. The analyst obtains from customers and/or users an initial requirements list.
2. For each requirements item, the analyst makes mapping from the item to concepts like Figure 6.
3. The analyst evaluates the quality of the requirements list by calculating the four measures mentioned in section 2.2. If the measures are sufficiently high to the analyst and the stakeholders, finish this procedure. Otherwise continue the steps below.
4. The analyst updates the requirements list according to the evaluation results of step 3.
   (a) When the completeness measure is low, the analyst should find and add new items to the requirements list.
      i. By focusing on the mapped concepts typed with "function", "object" and "environment" and by then tracing "apply" and "perform" relationships from the focused concepts in both directions, the analyst finds the concepts to be newly added.
      ii. By tracing "is-a", "has-a" and "require" relationships from the mapped concepts only in specified direction, the analyst finds the concepts to be newly added.
   (b) When the unambiguous measure is low, the analyst should find ambiguous items and then modify, remove or split them into several items. The analyst looks for a requirements item that is mapped to more than one "island" and it is a candidate of an ambiguous item. In the case of splitting it, the analyst does it into several items so that each of the items is mapped to only one "island".
   (c) When the correctness measure is low, the analyst focuses on the mapped concepts of the incorrect items, and asks stakeholders whether the items are really necessary or not.
   (d) When the consistency measure is low, one of two inconsistent items is selected to be removed. The inconsistent items can be detected by searching contradict relationships in the mapped elements. The priority relationships among the items can help the analyst and stakeholders to decide which item should be removed.
5. Back to the step 2.

Figure 7. A Procedure for Requirements Elicitation

specifications will be described, but how to tailor specification documents compliant to a standard such as IEEE 830 is out of scope of this procedure.

By using the models shown in Figure 6, we will illustrate the steps above. Our illustration process starts after finishing the steps 1 and 2, and the mapping has been already established as shown in Figure 6.

Results of Step 3

According to the definitions in section 2.2, we calculate the measures as follows.

(a) Completeness = 6/13 = 46 %
(b) Unambiguity = 1/3 = 33 % (Only an item #1 is unambiguous.)
(c) Correctness = 3/3 = 100 % (All items are mapped.)
(d) Consistency = 5/5 =100 % (No contradict relationship is contained.)

Note that # of requirements items is 3, # of all concepts is 13, # of mapped concepts is 6, # of all relationships is 15 and # of relationships between mapped concepts is 5 in Figure 6. As a result, the values of completeness and of unambiguity are low, thus we proceed to the next steps in order to improve them.

Results of Step 4

According to the results in Step 3, we update requirements items as follows. As a result, six candidates of new requirements items are found in Step 4.(a) and 4.(b), and completeness and unambiguity will be improved.

Step 4.(a) For completeness:

According to Step 4.(a).i, the following candidates are found: function(return) from the items #1 and #3, function(return), object(publication) from the item #2.
According to Step 4.(a).ii, the following candidates are found: object(member account) from the items #1, which is connected to the mapped concepts of item #1 through "require" relationship.
And then we compose the following four sentences from the newly found concepts "return", "publication" and "member account":

1. A copy of a book shall be returned.
2. Publications shall be added.
3. Member account shall be required when checking out.

4. The member shall have the member account.

Step 4.(b) For unambiguity:
According to Step 4.(b), the item #3 is mapped to two concepts object(a copy of a book) and object(a book); the former stands for a *physical* copy of a book and the latter denotes publication, which are connected through "has-a" relationship. These can be considered as "islands", and we should decide with which meaning the word "books" is used in the requirements list. We split the item #3 into the following two items according to the Step 4.(b), as follows, since the item #3 uses the word "books" with both of meanings.

5. Copies of a book shall be added.
6. New books shall be added.

There is an is-a relationship between object(a copy of a book) and quality(availability) mapped from the item #2. However, we do not apply Step 4.(b), because the concepts has the different types; object and quality.

Step 4.(c) For correctness:
We have to do nothing because the correctness measure shows 100%.

Step 4.(d) For consistency:
We also have to do nothing because the consistency measure shows 100%. However, we will have to do something in the next iteration because a contradict relationship can be introduced by the result in Step 4.(a) in this iteration.

Note that the completeness measure represents the ratio of mapped elements to all ontological elements in the thesaurus. It means that a requirements list refers to everything included in the thesaurus if the measure is 100%, and it is not so reasonable. In the above process, we had to exclude from the calculation of completeness the ontological elements that the analyst decided not to include in a next version of the requirements list in the Step 4.(a).

5 Case Studies and Discussion

To evaluate the user-friendliness and effectiveness of our method, we made a comparative experiment. In this section, we discuss the content of our experiment and its results.

5.1 Experiment

In our experiment, we had two types of subjects; *navigated subjects* and *free subjects*. The former subject should perform requirements elicitation tasks following our method, while the latter did not use the method but could do his/her task as he/her liked. Both types of subjects extended a requirements list by using domain ontology, and were instructed to record his/her process, especially extended requirements items and mappings between ontological concepts and the items. A navigated subject followed the procedure in section 4. A free subject was instructed to use the ontology as domain knowledge and to record the correspondence of the improved requirements items to ontological concepts during his/her requirements elicitation process and to use The free subject calculated the quality measures mentioned in section 2.2 after finishing his/her experiment. On the other hand, as for a navigated subject, they were being calculated by him/her during the experiment. By analyzing the numerical and process data resulted from the experiment, we discuss the user-friendliness and effectiveness of our method.

5.2 Experimental Results

We used a domain ontology of software music players, which have been constructed from many documents of software music players and it was shown in [9] The ontology consists of 48 concepts and 67 relationships. The following document (originally written in Japanese) was used as the initial requirements list.

1. Play a music, pause. Go to next or previous music.
2. Forward and rewind.
3. Change the speed of playing.
4. Adjust volume and mute.
5. Repeat play list.
6. Random play list.

We selected three undergraduate students in software engineering course, S1, S2, and S3 as our subjects. S1 and S2 were navigated subjects and S3 was a free subject. Since S1 and S2 followed our method, they iterated steps of the elicitation tasks twice, while S1 did not do but extended the requirements items all at once. Table 1 shows the overview of experimental results. For example, S1 started with decomposing 6 items in the initial list into 11 items, and after that, he/she mapped the 11 items into 14 concepts on our ontology. At this moment, the measures of correctness, completeness, consistency and unambiguity were 91%, 12%, impossible to calculate (the denominator was 0), and 100% respectively. In step 2, S1 had 24 items by adding 13 items and mapped concepts were increased to 21. This increase led to the improvement of completeness to 29% in S1's activity. Unlike S1 and S2, S3 did not decompose the initial sentences anymore.

5.3 Discussion

The findings obtained from the experimental results can be listed up as follows.

1. The number of mapped concepts per one requirements item reflects the semantic simplicity of the item. The numbers of subjects S1 and S2 (35/39=0.9, 26/24=1.1,

Table 1. Results of Experiments

Subject		Initial	Step 1	Step 2
S1	# Requirements Items	11	24	39
	# Mapped Concepts	14	21	35
	Correctness (%)	91	96	97
	Completeness (%)	12	29	46
	Consistency (%)	–	100	100
	Unambiguity (%)	100	50	49
S2	# Requirements Items	11	25	25
	# Mapped Concepts	13	26	26
	Correctness (%)	90	100	100
	Completeness (%)	11	22	22
	Consistency (%)	100	100	100
	Unambiguity (%)	80	100	100
S3	# Requirements Items	6	16	
	# Mapped Concepts	14	40	
	Correctness (%)	83	100	
	Completeness (%)	12	35	
	Consistency (%)	100	98	
	Unambiguity (%)	100	100	

after step 2) were relatively smaller than the number of S3 (40/16=2.5), and it means that the items of S1 and S2 are semantically simpler than S3's items. Therefore, our method seems to contribute to write simple requirements items. In fact, most items of S3 were complex sentences, each of which contained more than one requirement.

2. In the initial step, all subjects identified 13 or 14 mapped concepts. Therefore, mapping activities in our method seems to be objective. It can be considered the mapping results may be same whoever does this task of mapping.

3. Subject S1 and S2 iterated their processes twice according to our method. However, S3 did not iterate his process. Therefore, our method seems to encourage an analyst to iterate RE process.

4. In any subjects, the values of quality measures increased, and except for completeness values, the values became almost 100%. Therefore, our method seems to be effective to improve the quality of requirements items.

5. Subject S3 explored ontological concepts in breath-first way to make up for lacks of domain knowledge from the ontology, and especially, he focused on "apply" relationships in his process. Our method also encourages an analyst to focus on "apply", thus our method seems to be natural in the sense that it does not force analysts to perform constrained tasks unnatural to human.

6 Related Work

Roughly speaking, the other work related to our approach can be categorized into two groups; one is the technique to develop domain ontologies and another is the application of ontological technique to requirements engineering. In research community of ontology, many computerized tools to develop ontologies [2, 3] have been developed, and some electronic versions of thesauruses such as WordNet [4] can be found. DAML Ontology Library [1] provides ontologies for various domains and they are described in standard OWL based language so as to achieve high interoperability. We can use them for our method by extracting sub-ontologies from them and by integrating the several ontologies. Furthermore many persons participate in developing ontologies of their own and upload them to the library. As a result, it is incrementally extended day by day and thus we can get up-to-date ontologies of higher quality.

In [12], a domain ontology can be captured as domain experts' knowledge and a case study to try to extract an ontology from requirements documents is presented. Although the aim of these contributions is different from our paper, they are very useful to construct domain ontologies of high quality with less human efforts. According to [5], a domain ontology is one of the crucial issues on developing Semantic Web based systems and the techniques of requirements engineering can be applied to the development of an ontology.

The studies in the second category are similar to our work from the viewpoint of their goals. In [7], the idea to use a domain model represented with RML for expressing the meaning of requirements was firstly introduced. However, it did not discuss the technique to improve the quality of requirements so much. LEL (Language Extended Lexicon) [5] 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 methods as guidance procedures nor semantic inference mechanisms. PAORE [11] is a method for refining requirements by using a domain thesaurus so as to select software packages, but its thesaurus has no inference mechanisms. It was impossible to detect inferentially missing requirements and inconsistent ones by this method. A feature diagram in Feature Oriented Domain Analysis [10] can be considered as a kind of representations of a domain thesaurus, and in [22], a technique to analyze semantic dependencies among requirements by using features and their dependency relationships was proposed. The idea of this technique is similar to our approach in the sense that requirements can be semantically analyzed. However, the aim of our approach is the support for requirements elicitation, while [22] just aimed at modeling semantic dependencies lying behind a requirements specification. In [16], the authors combined several formal methods by using ontology as their common concepts. This is another application of ontology in requirements engineering, especially method integration, but its goal is different from ours.

Another remarkable work related to this area was Requirements apprentice (RA) [17]. It used reusable templates

called Cliche to assist a requirements analyst in creating and modifying requirements, and the cliches provided common forms of requirements specification in a specific domain. Using metrics for assisting an analyst is one of the key differences between our work and RA. Spanoudakis et al [20] proposed similarity metrics between software artifacts. Our metrics can be regarded as some kinds of similarities between a requirements list and an ontology, thus we can extend or improve our metrics by using its technique.

7 Conclusion

In this paper, we propose a method for requirements elicitation by using ontology. The method helps a requirements analyst to extend requirements systematically by taking account of the semantic aspect of requirements. We define logical structures of artifacts during the method and its procedure. In the procedure, the following two kinds of activities are iterated; evaluation of requirements by using quality metrics, and revision of the requirements based on the structural characteristics in ontology. We partially assess the user-friendliness and effectiveness of this method through an experiment. However, our experiment mentioned in section 5 was too small to argue the generality of the experimental findings.

To perform the method efficiently and effectively, CASE tool(s) must be provided. We will develop such CASE tools by using inference systems such as prolog [21] because inference is one of the important factor in our method. Such CASE tools will enable us to have much more experimental findings. For example, we have already had ideas to predict and guide requirements changes [9] and we will be able to assess our ideas by using CASE tools. The result of our method heavily depends on the quality of domain ontologies, thus acquiring good ontology data is important issue in practice. Results of the related works in the previous section are of course used to acquire ontologies for our method. In addition, we are going to explore the possibility to acquire ontology data from existing documents such as manuals and help files of software products by using text-mining and lexical knowledge acquisition technology [14]. In fact, we are starting to develop automated techniques to extract thesauruses from natural-language documents [15].

References

[1] DAML Ontology Library. http://www.daml.org/ontologies/.

[2] KAON Tool Suite. http://kaon.semanticweb.org/.

[3] Proc. of 2nd International Workshop on Evaluation of Ontology-based Tools. http://sunsite.informatik.rwth-aachen.de/Publications/CEUR-WS//Vol-87/.

[4] WordNet: A Lexical Database for the English Language. http://wordnet.princeton.edu/.

[5] K. Breitman and J.C.S.P. Leite. Ontology as a Requirements Engineering Product. In *Proc. of RE03*, pages 309–319, 2003.

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

[7] Sol J. Greenspan, John Mylopoulos, and Alex Borgida. On formal requirements modeling languages: RML revisited. In *16th ICSE*, pages 135 – 147, 1994.

[8] M. Gruninger and J. Lee. Ontology: Applications and Design. *Commun. ACM*, 45(2), 2002.

[9] Haruhiko Kaiya and Motoshi Saeki. Ontology Based Requirements Analysis: Lightweight Semantic Processing Approach. *Proc. of QSIC2005*, pages 223–230, Sep. 2005.

[10] K. Kang, J. Lee, and P. Donohoe. Feature-Oriented Product Line Engineering. *IEEE Software*, 19(4):58 – 65, 2002.

[11] J. Kato, M. Saeki, A. Ohnishi, N. Nagata, H. Kaiya, S. Komiya, S. Yamamoto, H. Horai, and K. Watahiki. PAORE: Package Oriented Requirements Elicitation. In *Proc. of APSEC2003*, pages 17 – 26, 2003.

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

[13] A. Maedche. *Ontology Learning for the Semantic Web*. Kluwer Academic Publishers, 2002.

[14] R. Mitkov, editor. *The Oxford Handbook of Computational Linguistics*. Oxford University Press, 2003.

[15] A. Miura and M. Saeki. A Technique for Constructing Domain Thesauruses from Co-occurrence Information of Words. In *Proc. of LKR 2006*, pages 23–26, 2006.

[16] M. Petit and E. Dubois. Defining an Ontology for the Formal Requirements Engineering of Manufacturing Systems. In *Proc. of ICEIMT'97*, 1997.

[17] H.B. Reubenstein and R.C. Waters. The Requirements Apprentice: Automated Assistance for Requirements Acquisition. *IEEE Trans. on Software Eng.*, 17(3):226–240, Mar. 1991.

[18] M. Saeki. Ontology-Based Software Development Techniques. *ERCIM News: http://www.ercim.org/publication/Ercim_News/enw58/saeki.html*, 58, 2004.

[19] M. Saeki, H. Horai, and H. Enomoto. Software Development Process from Natural Language Specification. In *Proc. of 11th ICSE*, pages 64–73, 1989.

[20] G. Spanoudakis and P. Constantopoulos. Measuring Similarity Between Software Artifacts. In *Proceedings of SEKE '94*, pages 387–394, June 1994.

[21] tuProlog. http://lia.deis.unibo.it/research/tuprolog/.

[22] W. Zhang, H. Mei, and H. Zhao. A Feature-Oriented Approach to Modeling Requirements Dependencies. In *Proc. of RE'05*, pages 273–284, 2005.