Extracting Conceptual Graphs from Japanese Documents for Software Requirements Modeling

Ryo Hasegawa[1] **Motohiro Kitamura**[1] **Haruhiko Kaiya**[2] **Motoshi Saeki**[1]

[1]Dept. of Computer Science, Tokyo Institute of Technology
Ookayama 2-12-1, Meguro-ku, Tokyo 152, Japan
Email: saeki@se.cs.titech.ac.jp
[2]Dept. of Computer Science, Shinshu University
Wakasato 4-17-1, Nagano 380-8553, Japan
Email: kaiya@cs.shinshu-u.ac.jp

Abstract

A requirements analysis step plays a significant role on the development of information systems, and in this step we produce various kinds of abstract models of the systems (called requirements models) according to the adopted development processes, e.g. class diagrams in the case of adopting object-oriented development. However, constructing these models of sufficient quality requires highest intellectual tasks and skills of human requirements analysts. In this paper, we develop a computerized tool to extract from a set of Japanese text documents conceptual information, called *conceptual graph*, which can be used as intermediate representation to generate software requirements models. More concretely, by applying the variation of text-mining techniques that we have developed, we extract significant words from text documents referring to the same problem domain and identify relevant relationships among them. The extracted words can be considered as concepts and they are constituents of a conceptual graph in the domain. This constructed graph can be used for generating requirements models, e.g. object oriented models, feature model, and even as a domain ontology that can be utilized during requirements analysis activities. We have made experimental analyses of our tool. This paper also includes the discussion on how the extracted conceptual graph can act as an object-oriented model, a feature model and a domain ontology, in order to show its wide applicability.

Keywords: Conceptual Graph, Requirements Modeling, Text mining, NL processing

1 Introduction

Since a requirements analysis step is the first one in information systems development processes, the quality of the artifacts that are produced in this step greatly affects on the quality of a final artifact, i.e. an information system. If we constructed an artifact of lower quality in this step, for example an incomplete and/or inconsistent one, we might re-do our activities after completing the final artifact and as a result we might spend much effort and the development cost might exceed an estimated budget.

In this requirement analysis step, we produce abstract models of the information system according to the adopted development process style. For example, when we use object-oriented (OO) development process, we produce a class diagram as an object-oriented model. If we develop a product belonging to a certain family and adopt Feature-Oriented Analysis technique, we should produce a feature oriented model. Thus we can produce various kinds of model in a requirements analysis step according to the adopted development process. We call these models, i.e. abstract models of the system that produced in a requirements analysis step, *requirements models*. We should construct a requirements model of high quality as early as possible to reduce development costs and efforts. However, human engineers are required to perform highly intellectual and complicated activities and to have distinguished skills in order to construct a requirements model of high quality. In addition, they should be experts to various modeling techniques that can be adopted. A current status is that a limited number of domain experts are involved in requirements modeling in their domains, spending their large efforts. We need some supporting techniques to assist human engineers in constructing various types of requirements models of higher quality with less effort.

On the other hand, it is a rare case that we construct a requirements model whose domain is quite new and does not appear before. If we had reusable assets helpful for requirements modeling, we could get the model efficiently. However, we have not accumulated sufficient reusable assets of requirements models in a certain domain yet. Rather, we can get many text documents referring to the domain, including the electronic texts lying over Internet. In fact, the experts to modeling frequently use the documents regarding to the topics relevant to the problem domain so as to get important information. Thus, it can be considered as a promising support technique to extract from the documents information necessary for requirements modeling. These documents are written in natural language, and the constituents that a requirements model should have, e.g. concepts and their relationships appear in the documents as words and their co-occurrences in a suitable abstraction level, because of the abstractness of natural language descriptions. The words that commonly appear in the documents of a domain, except for general words such as be-verbs, prepositions, particle, etc., can be considered as the representation of significant concepts in the domain. In addition, the usages of theses words such as co-occurrence and modification relationships suggest the relationships between the concepts that the words denote. Thus we focus on the extraction of these words and relationships from the documents.

To generate various kind of requirements model, we extract an intermediate representation from a set of text documents by using the combination of natural-language (NL) processing and text-mining techniques so that it can be (semi-)automatically transformed into various requirements models. Our intermediate representation is called *conceptual graph*, which includes concepts and their relationships extracted from the documents. Furthermore requirements analysts can use this graph to make up for their lack of domain knowledge during their requirements elicitation activities. Figure 1 shows the overview of our

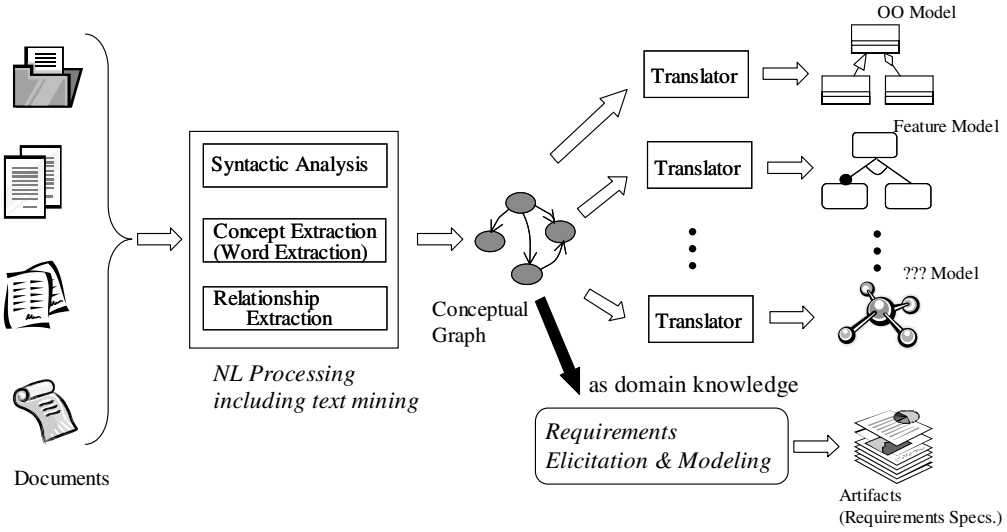


Figure 1: Overview of Our Approach

approach. In this paper, we have developed a computerized tool for extracting conceptual graphs from a set of documents. As will be discussed in the section of Related Work, we can find many studies to extract specific requirements models such as OO models from a *single* document. Unlike these studies, we use a set of documents as inputs so that we can get stable and reusable conceptual information. It is very significant which information we should extract from documents. Since our target is information systems, we adopt the concepts and their relationships that we frequently use in modeling them, e.g. Object, Function, Is-a relationship (generalization) and Has-a (aggregation), etc., and construct from them a meta model of the conceptual graphs.

The rest of this paper is organized as follows. In the next section, we explain the basic idea and show the logical structure of the conceptual graphs, i.e. meta model. We extract from Japanese documents information based on this meta model. Section 3 presents the process for extracting conceptual graphs and the computerized tool using NL processing and the text mining technique that we have proposed. Since our conceptual graphs have more specific conceptual types and relationship ones rather than usual thesauruses, we should develop newly a text mining technique. Section 4 includes experimental results on the effectiveness of our developed tool. In section 5, we discuss how to get software requirements models from the constructed conceptual graph in order to show its wide applicability. In sections 6 and 7, we discuss related work and our current conclusions together with future work, respectively.

2 Meta Model of Conceptual Graphs

2.1 Requirements to a Meta Model

As mentioned in section 1, we have a variety of notations for requirements models such as Entity Relationship Diagram and UML (Class Diagram etc.), and they have different meta concepts for description. In the case of Class Diagram, it has meta concepts Class, Attribute, Operation, Association, etc. Therefore, we need to clarify the structure of conceptual graphs, i.e. a meta model of conceptual graphs so that we can extract Classes, Attributes, Operations, Associations etc. from the conceptual graph afterward. Our meta model should 1) have extensive meta concepts so that we can derive various requirements models, even reusable assets such as feature model of FODA [1], from a conceptual graph that is its instance, 2) have useful meta concepts specific to the area of information system, and 3) be based on the information that can be automatically gathered from text documents.

2.2 Meta Model of Conceptual Models

In order to satisfy the requirements to the meta model mentioned in section 2.1, we analyzed the existing software requirements modeling methods, referring to UML's meta model [2], Method Engineering meta model [3], Method Common Meta Model [4], etc. and have got the meta model shown in Figure 2. Our meta model consists of concepts and relationships among the concepts, and it has several subclasses of "concept" class and "relationship". In the figure, "object" is a subclass of a concept class and a relationship "apply" can connect two concepts. Concepts and relationships in Figure 2 are adopted so as to easily represent the semantics in information systems. Intuitively speaking, the concepts "object", "function", "environment" and their subclasses are used to represent functional aspects of the systems. On the other hand, the concepts "constraint" and "quality" are used to represent non-functional aspects. The concept "constraint" is useful to represent numerical ranges, e.g., speed, distance, time expiration, weight and so on.

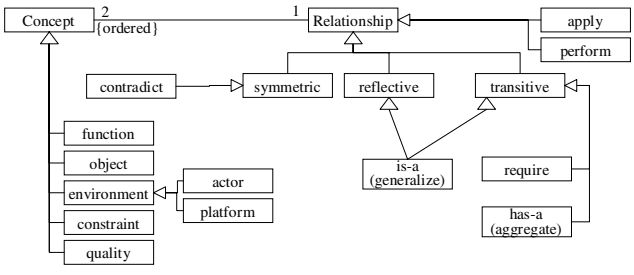


Figure 2: Meta Model of Conceptual Graphs

Figure 3 shows an example of a conceptual graph of the problem domain of "making estimates", an instance of the meta model of Figure 2, which is depicted in the form of Class Diagram, and it is a screenshot of our tool. Note that our tool is for Japanese only and the figures of tool screens have been produced by translating Japanese words into English directly. A concept and its type are depicted as Class and a stereo type respectively in the figure. Readers can find the concepts of type object, "esti-

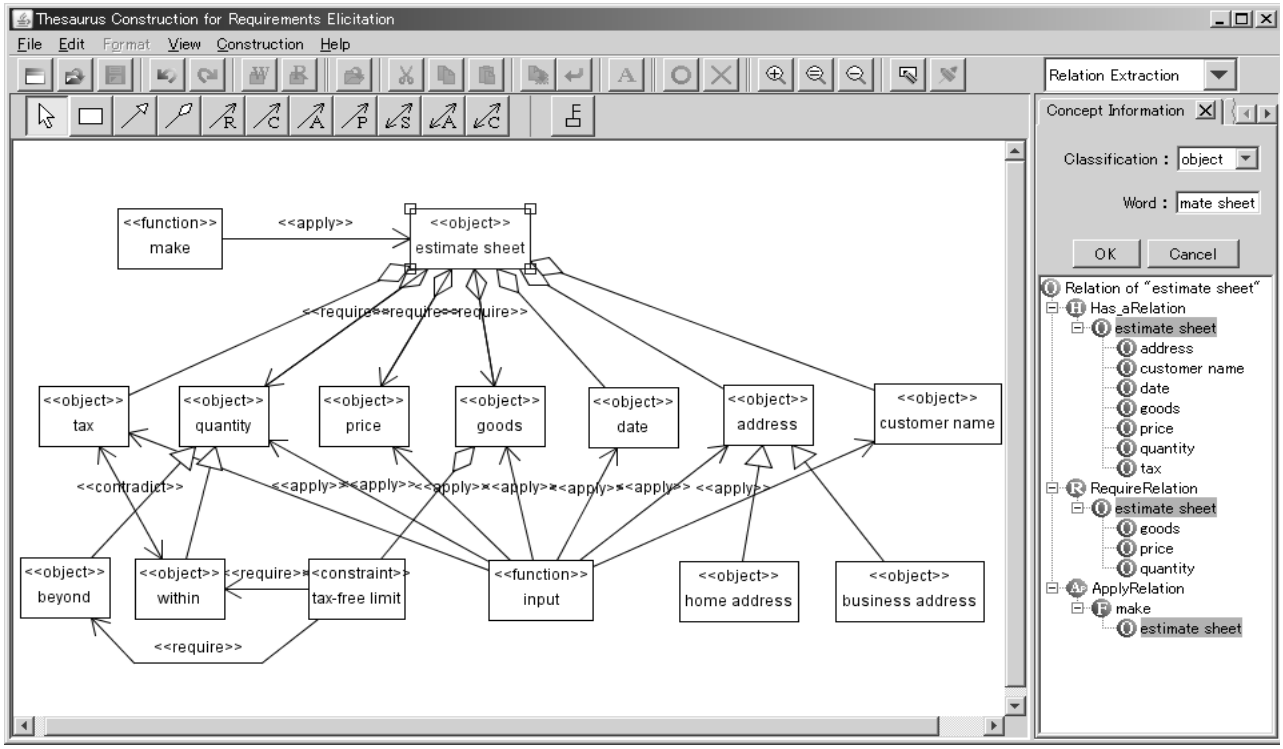


Figure 3: A Tool Screenshot of Relationship Extraction: An Example of a Conceptual Graph

mate sheet", "goods", "tax", etc, and "input" and "make" of type function. There are two relationships between "estimate sheet" and "goods"; one is the relationship of type "require" and another is of "has-a" [1]. Since an estimate sheet should have some columns on goods, their prices and their quantity information, we use the combination of these two types (require and has-a) of relationships between the estimate sheet and them. The concept "input" of type "function" is applied to "goods", "quantity", "prices" etc. in order to input these data, and thus we can have the relationships of type "apply" to them.

3 A Supporting Tool for Extracting Conceptual Graphs

In this section, we focus on the technique to extract constituents of a conceptual graph from Japanese text documents. The quality of a conceptual graph greatly depends on the quality of used text documents. If we use a document of lower quality, we also get a graph of lower quality. There are no formal techniques to validate the quality of the extracted conceptual graph. We consider that the quality of the conceptual graph can be validated by *social consensus* of domain experts and by its usability to our applications. We can consider that concepts and relationships commonly appearing in many documents on a domain have established social consensus. The larger the number of documents is, the higher the quality of the extracted graph can be, because the concepts appearing in the many documents are widely accepted in this domain. As for the usability to our applications, we will discuss it in section 5.

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 extracting a conceptual graph are 1) Word extraction for extracting from documents the important words that can be considered as useful concepts and 2) Relationship extraction for discovering the relationships among the extracted words, as shown in Figure 4.

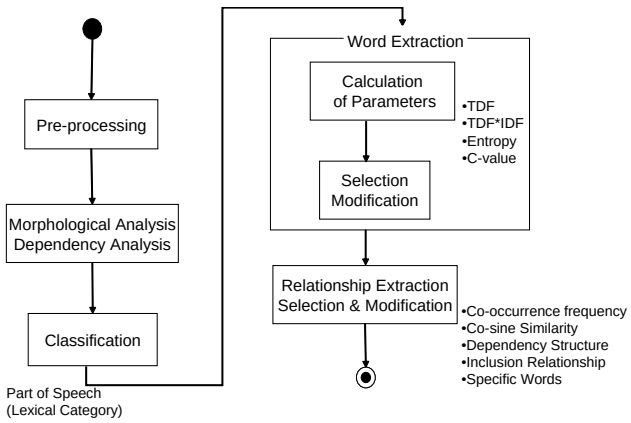


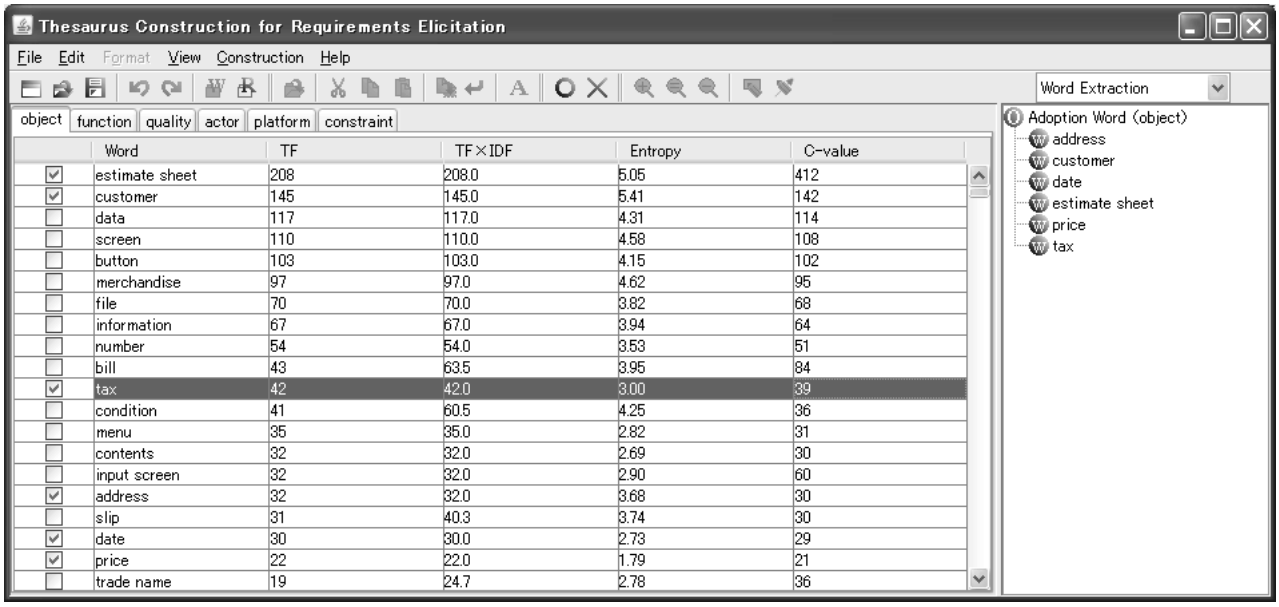
Figure 4: Process for Extracting a Conceptual Graph

3.1 Word Extraction

After morphological analysis and dependency analysis, we identify part-of-speech categories of the meaningful words appearing in the documents such as noun, verb, adjective etc. These steps can be performed automatically using the natural-language processing tool called Cabocha (dependency structure analyzer for Japanese)[2]. By using part-of-speech information of words, we classify the words into the types of the concepts shown in Figure 2 such as object, function and quality. For example, "estimate sheet" is a noun and is classified into an object concept. In the next step, our tool calculates various measure

[1] We use the same notation as UML Class diagram to represent "has-a", i.e. aggregation relationship.

[2] http://chasen.org/taku/software/cabocha/



	Word	TF	TF×IDF	Entropy	C-value
☑	estimate sheet	208	208.0	5.05	412
☑	customer	145	145.0	5.41	142
☐	data	117	117.0	4.31	114
☐	screen	110	110.0	4.58	108
☐	button	103	103.0	4.15	102
☐	merchandise	97	97.0	4.62	95
☐	file	70	70.0	3.82	68
☐	information	67	67.0	3.94	64
☐	number	54	54.0	3.53	51
☐	bill	43	63.5	3.95	84
☑	tax	42	42.0	3.00	39
☐	condition	41	60.5	4.25	36
☐	menu	35	35.0	2.82	31
☐	contents	32	32.0	2.69	30
☐	input screen	32	32.0	2.90	60
☑	address	32	32.0	3.68	30
☐	slip	31	40.3	3.74	30
☑	date	30	30.0	2.73	29
☑	price	22	22.0	1.79	21
☐	trade name	19	24.7	2.78	36

Figure 5: A Tool Screenshot of Word Extraction

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. The importance degree results from the number of the documents the word appears.
3. Entropy: logarithmic value of the term frequency of a word weighted with its entropy [5]. Intuitively speaking, an entropy value of the word A comes to be lower if A appears uniformly throughout all documents.
4. C-value : the term frequency of a word weighted with its length and its occurrences as a part of multi-words. This value is for the characteristic of Japanese texts that they frequently include many occurrences of multi-words. A multi-word is a combination of several words.

Figure 5 shows an example of the result of word extraction. As shown in the figure, the words are measured and sorted in descending order of the measure values. A user of the tool can select the important words denoting concepts in a conceptual graph of a problem domain, by checking boxes on the sorted list of the measured words. In the example of the figure, the user has manually selected the words "estimate sheet", "customer", "tax", "address", "date" and "price".

3.2 Relationship Extraction

After selecting the words, the user proceeds to the step of relationship extraction. As shown in Figure 4, the tool calculates the number of times a pair of words included in a sentence in the documents, i.e. co-occurrence frequency (CF) of two words and cosine similarity (CS) of the frequency of co-occurrence vectors, in order to find the semantically relevant word pairs. If the two words co-occur frequently, we can consider the two concepts denoted by them are semantically related to each other.

The calculation method of cosine similarity of co-occurrence vectors is as follows. Suppose that the words $y_1, \cdots, y_n$ frequently co-occur with the word x in the documents. We can define a co-occurrence vector $\mathcal{V}_x$ of the word x as $(c(x, y_1), \cdots, c(x, y_n))$ where $c(x, y_i)$ is the number of times in which the words x and y_i co-occur. Thus we can calculate cosine similarity (CS) of the co-occurrence vectors of the words u and w as $(\mathcal{V}_u \cdot \mathcal{V}_w)/(|\mathcal{V}_u| \cdot |\mathcal{V}_w|)$. If the cosine similarity is sufficiently higher, we can consider that the words u and w are used in a similar way in the documents and that they have a certain semantic relationship.

After calculating CFs and CSs, pairs of words whose CF and CS are higher than certain thresholds are basically selected as candidates of the relationships to be included in a conceptual graph. Based on types of words (e.g. object, function and quality) and dependency structures in the sentences, the tool suggests the types of the concept relationships. Figure 6 shows the detailed process to extract relationships and to identify their types. Suppose that we focus on the words A and B, as shown in the figure. If A and B co-occur in a sentence and they also co-occur with a specific word such as "require", "contradict", "such as", etc., we decide that A and B has a relationship of types "require", "contradict" or "is-a", respectively. If A is a precisely right-side substring of B, we consider the relationship as "is-a". For example, the word "new estimate" (shinki-mitsumori-sho in Japanese) *is an* "estimate" (mitsumori-sho), because the word mitsumori-sho appears in shinki-mitsumori-sho as its right-side substring. If the CF value of A and B is higher, we check the dependency structure of the sentences where they co-occur and by their types and their syntactic roles such as subject and object etc., we decide the type of their relationship. For example, suppose that the types of A and B are "object" and "function" respectively. In addition, if A is an object in grammatical sense and B is its verb in a sentence, the tool suggests an "apply" relationship between A and B. "Apply" relationship between object A and function B means that the function B is applied to the object A. CS values are used to detect require and has-a relationships.

Figure 3 shows an example of the detected concepts and their relationships 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 conceptual graph.

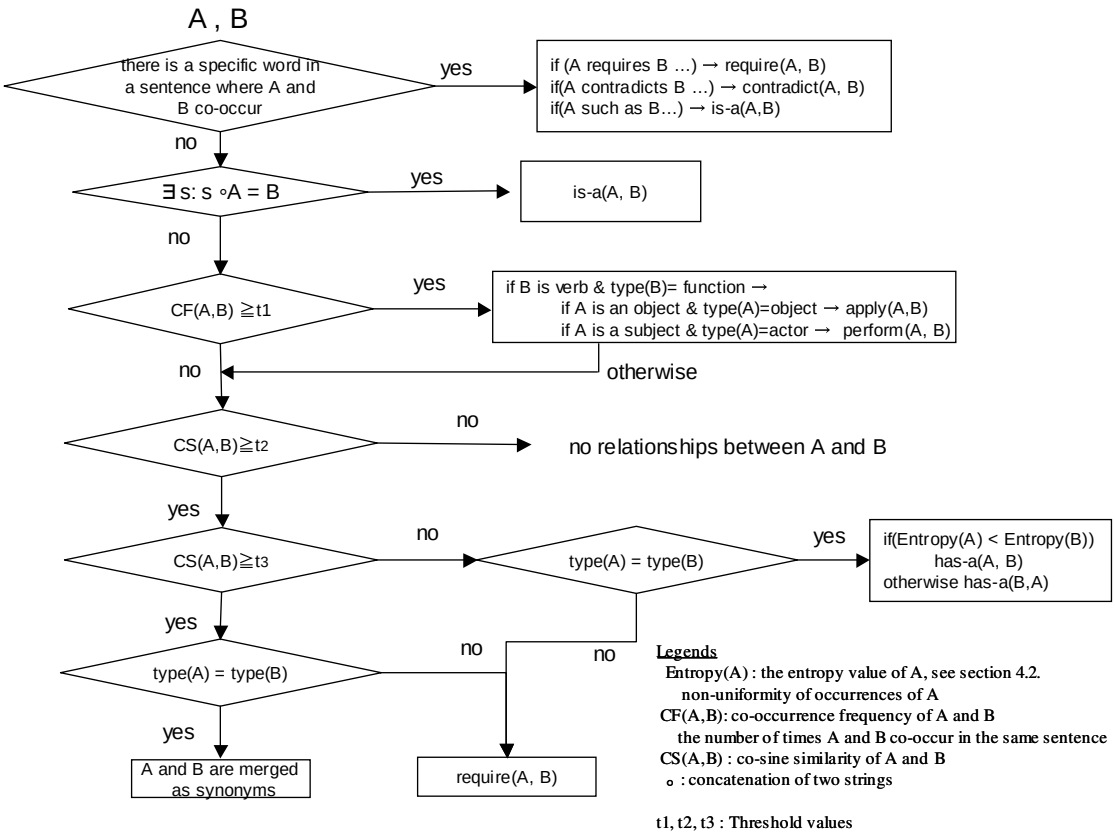


Figure 6: Relationship Extraction Process

4 Experimental Results

To assess our tool, we made several experiments, and in this section we discuss these experimental results.

4.1 Aims of Our Experiments

The essential aim of our experiments is to show that our tool allows any requirements analyst, who has a skill and domain knowledge in a certain level, to derive efficiently conceptual graphs of high quality. For these experiments, we had several subjects who had experiences in software development of more than 5 years including requirements analysis and software design. They had also actually developed software of the problem domains that we adopted in the experiments. Thus we can consider all of them had skills of requirements modeling and domain knowledge in a certain level. And we set threshold values of t1, t2 and t3 of Figure 6 to 3, 0.75 and 0.9 respectively.

We can decompose the above aim into the following items;

1. Any analyst[3] can construct conceptual graphs efficiently. Basically, we observe how long it took our subjects to complete their conceptual graphs.

2. Any analyst can get the same results, i.e. the same conceptual graphs, if they use the same documents as inputs to the tool. We have two subjects having the same skills and knowledge, more concretely having similar experiences, and make them construct conceptual graphs from the same documents by using our tool. After their constructing graphs, we compare their results and check how many parts of their constructed conceptual graphs are the same.

[3]In the context of this section, as mentioned above, "analysts" have sufficient skills, experience and domain knowledge like our subjects.

Table 1: Results on Feed Readers and POS systems

Problem Domain	Spent Time	Concepts	Relationships
Feed Reader	180 min.	178	270
POS System	160 min.	226	252

3. Any analyst can construct conceptual graphs of high quality. In fact, it is difficult to measure the quality of a conceptual graph. Thus we pay attention to the following two points to estimate the quality of conceptual graphs;

   (a) how many constituents of her graph the subject should modify so as to get to the graph at the quality level that she could be satisfied.

   (b) whether the conceptual graph that the subject constructed could be transformed to requirements models and be used as domain knowledge in requirements elicitation processes.

   The second point is related to the application of conceptual graphs and it is very significant to show that they graph can be used for requirements analysis tasks. This point will be discussed in the section 5.

4.2 Spent Time for Constructing Conceptual Graphs

We picked up specific problem domains and investigated how long and how large our subjects constructed conceptual graphs, in order to show that they could do efficiently. We selected the two domain Feed Reader and POS (Point of Sales) systems. Their results are shown in Table 1. For example, the subject of Feed Reader finally constructed the graph having 178 concepts and 270 relationships in 180 minutes. In this experiment, we gave 17 documents

Table 2: Results on Making Estimates

Spent Time	Concepts	Relationships
260 min.	218	432
180 min.	201	363

Table 3: Results on a Record Management System

	Concepts	Relationships
Used without any modifications	64 (86.4%)	76 (36.7%)
Modified	4 (5.4%)	62 (30.6%)
Added	6 (8.1%)	64 (31.6%)
Total number	74 (100%)	202 (100%)

for the subject of Feed Reader and 14 documents for the subject of POS system. The lengths of the documents that we used were from 3 to 23 pages of A4 paper size. Although their tasks included manual activities to modify the graphs, we consider that our tool is helpful to construct conceptual graphs of practical size within reasonable labor time. In addition, our subjects pointed out that they could know which parts they had to concentrate on for their understanding because our tool suggested significant words in these domains.

4.3 Similarity of the Constructed Graphs

We had two subjects and each of them developed a conceptual graph using our tool from the 8 documents referring to "making estimates" domain. The result is shown in Table 2. Although our two subjects spent different time (260 and 180 minutes respectively) in constructing their conceptual graphs, the sizes of the graphs were similar. They extracted 218 and 201 concepts respectively as shown in the table, and 147 of them were quite the same. Thus about 70% of the extracted concepts were commonly included in the graphs that different persons constructed, and we consider that any analyst can reasonably construct a conceptual graph at a certain level.

4.4 Quality of Conceptual Graphs: Modification Efforts

In the third experiment, we investigated the quality of the constructed conceptual graphs by measuring how much effort the subjects should modify manually the graphs that the tool derived. We selected a domain of "a record management system in a school (for storing and managing records of students' scores and credits)". We used 8 manuals for existing software for record management systems. Our subject, a skilled domain expert created a conceptual graph for "a record management system in a school" by using our tool. As shown in Table 3, he finally got 74 concepts and 202 relationships and these can be considered as the graph of high quality, because the distinguished expert manually modified and completed the graph. The tool automatically extracted 68 (64 + 4) and 64 of 68 were used without any modifications. 4 of 68 extracted concepts were modified and 6 concepts were newly added by the expert. As for the relationships, the tool automatically recognized 76 + 62 relationships and 76 were used without any modifications. From this table, our tool could create totally more than 60% of concepts/relationships in the graph. In almost of 62 modifications of the relationships, the expert manually modified has-a and is-a relationships to require relationship, because our technique of Figure 6 could not distinguish correctly require relationship from has-a and is-a relationships. From this experiment, although the tool could not necessarily extract the conceptual relationships accurately, it could do concepts satisfactorily. Human efforts were necessary to get more complete relationships. However, the time spent in modifying and adding concepts and their relationships was less than 2 hours and thus in a tolerable range.

Note that the goal of this tool is not to automate completely the creation of a precisely correct conceptual graph, but to support human activities and produce a useful graph for our application. In the application of modeling the requirements of an information system, it is more important to include concepts and relationships as many as possible, in order to avoid lacking requirements. Thus our tool tries to show many candidates of concepts and relationships. By using our tool, a requirements analyst selects appropriate ones out of the candidates, and replaces their types into correct ones if their types are inappropriate.

5 Applications of a Conceptual Graph

In this section, to show the wide applicability of the conceptual graphs constructed using our approach, we explain how to derive an object oriented model and a feature model from an extracted graph. And we show another application where the conceptual graph can be used as domain knowledge for software requirements elicitation processes. By showing the wider applicability of the constructed graph, we can estimate its quality.

5.1 Transforming a Graph into an Object Oriented Model

One of the most popular modeling techniques in software engineering is object oriented modeling and we use class diagrams to represent them. As shown in Figure 2, our conceptual graphs are based on an object oriented modeling technique. Therefore, an object oriented model can be derived from our conceptual graph straightforward. The outline of derivation rules is as follows. For simplicity, we call each subclass of a concept or a relationship in Figure 2 as XX-concept or YY-relationship. For example, we call "function" subclass of a concept "function-concept". An object-concept in our conceptual graph corresponds to a class in an object oriented model, and function-concepts related to the object-concept with an apply-relationship become methods of the class. Constraint-concepts related to the object-concept become attributes of the class. Is-a-relationships and has-a-relationships in our conceptual graph simply correspond to inheritance and aggregation relationships in the object oriented model.

Figure 7 shows a class diagram (an object oriented model) derived from a conceptual graph in Figure 3 by using the above rules. Object-concepts such as "estimate sheet", "goods" and "price" become classes in the class diagram, and has-a-relationships in the conceptual graph become aggregation relationships. An aggregation relationship between "estimate sheet" and "goods" is a typical example in Figure 7. A function-concept "input" in the conceptual graph has apply-relationships to several object-concepts as shown in Figure 3. Therefore, classes corresponding to the object-concepts in the class diagram has a "input" method as shown in Figure 7. For example, a class "goods" has a method "input".

5.2 Transforming a Graph into a Feature Model

Feature modeling was developed by Kang et al, and reusable assets in a product line development can be represented in the model. A definition of a feature is given in [1] as "a prominent or distinctive user-visible aspect, quality or characteristic of a software system or a system". A feature model has a hierarchical (normally tree) structure among features, which are inherent concepts of a product family, and it is normally depicted in the tree-like diagram

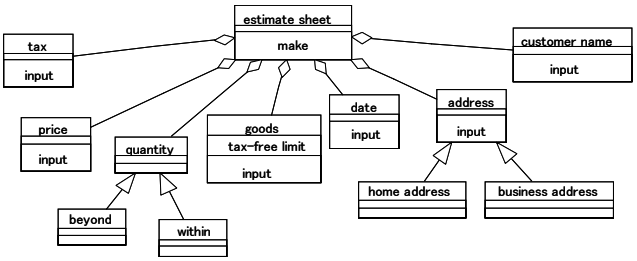


Figure 7: Deriving a Class Diagram

called feature diagram. To specify a model of a product in a product family, features in a feature diagram are chosen in a top down manner, i.e., sub-features are chosen after their super-feature was chosen. When a sub-feature has a mandatory relationship to its super-feature, this sub-feature should be chosen, i.e. the product should have this sub-feature. There are other kinds of relationships among features such as optional, alternative, exclusive and so on.

A conceptual graph can be derived from documents about a product family, and concepts in the graph correspond to features in a feature model. Has-a-, is-a-, apply- and perform-relationships in the graph correspond to relationships between super- and sub-feature relationships. In the case of is-a-relationship, the derived relationship between a super- and a sub-feature should be alternative relationships. In addition, since the properties of a super concept are inherited to its sub concepts in our conceptual graph, we consider that the concepts related to the super-concept would be also related to all of its sub concepts.

In Figure 8, we show a feature diagram derived from the conceptual graph in Figure 3. All concepts in the graph are transformed into features at first. Since the features such as "make", "tax" and "goods" have "has-a", "is-a" or "apply" relationships with a feature "estimate sheet", these features become sub-features of the feature "estimate sheet". A sub-feature "goods" is a mandatory feature of its super-feature "estimate sheet" because "goods" has a "require" relationship to its super-feature "estimate sheet" as well as a "has-a" relationship. Suppose that an apply-relationship between a function-concept and an object-concept is included in our conceptual model. The function-concept corresponds to a sub-feature of a feature corresponding to the object-concept. In addition, this sub-feature becomes a mandatory feature if the apply-relationship is only one between the function and the object, because it is the only one function that can manipulate the object. A sub-feature "make" in Figure 8 is a typical example of this kind of mandatory features and "make" is the only one that can manipulate "estimate sheet" according to the conceptual graph shown in Figure 3. As shown in the figure, object-concepts "home address" and "business address" are the sub-classes of an object-concept "address" because these concepts have is-a relationships to "address". These two concepts become alternative sub-features of a feature "address" as shown in Figure 8. In addition, these two features has a sub-feature "input", because a function-concept "input" has an apply-relationship to the object-concept "address" and two concepts "home address" and "business address" are the sub-classes of the object-concept. By applying these kinds of transformations, the feature diagram of Figure 8 can be derived from Figure 3.

In a feature diagram, several kinds of relationships among features, e.g., a dependency relationship among features and an exclusive relationship between features, are allowed in addition to the tree-like hierarchy of features. When there is only a require-relationship between two concepts in our conceptual graph, we have a dependency relationship between the two features corresponding to these concepts. When there is a contradict-relationship among concepts in a conceptual graph, there is an exclusive relationship between the features corresponding to the concepts. An example of exclusive relationships appears between features "tax" and "within" in Figure 8. An example of dependency relationships appears between features "tax-free limit" and "within".

In deriving object oriented models and feature models, their derivation rules can be formally defined, and these derivations can be automatically achieved. However, the quality of derived models cannot be guaranteed without the inspection of human experts. Thus such derivation rules play a role of guidelines only. This kind of derivation should be achieved interactively, and the finally derived models should be improved by manual.

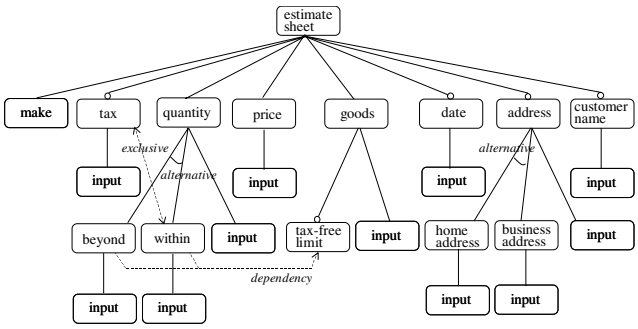


Figure 8: Deriving a Feature Diagram

5.3 Using as a Domain Ontology

Knowledge on a problem domain where an information system is operated (simply, domain knowledge) plays an important role on eliciting system 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 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. Although interviews with domain experts are one of the solutions to avoid this problematic situation, communication gaps between the analysts and the domain experts resulted from their knowledge gaps [6]. 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.

We have proposed how to use domain ontologies for requirements elicitation [7] where domain ontologies are used to make up domain knowledge to requirements analysts during requirement elicitation processes. In this framework, how to create domain ontologies of high quality efficiently is a crucial issue. Our tool for extracting conceptual graphs can be used to create domain ontologies for supporting requirements elicitation processes.

In this section, we present the basic idea how to use our conceptual graph as domain knowledge to detect lacking requirements and inconsistent requirements. Below, let's consider how a requirements analyst uses a conceptual graph of a certain domain for completing requirements elicitation. Suppose that a requirements document initially submitted by a customer is itemized as a list. At first, an analyst should map a requirement item (statement) in a requirement document into concepts of the conceptual graph as shown in Figure 9. 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 graph suggests to the analyst what part she 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 has a require-relationship to A in the conceptual graph G. The inference resulted from "C has a require-relationship to A (i.e. 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. The details of this technique are out of scope of this paper, and the readers who have a great interest to it can see [7].

To assess this technique, we used the conceptual graph of Feed Reader in section 4.1 and made comparative experiments of requirements elicitation of a specific feed reader system. As a result, subjects with less domain knowledge could get the same results as a domain expert, more concretely they could elicit requirement of the same quality as the domain expert did. The details of the experiments and their results are shown in [8]. This result means that our conceptual graph is applicable as domain knowledge for requirements elicitation processes.

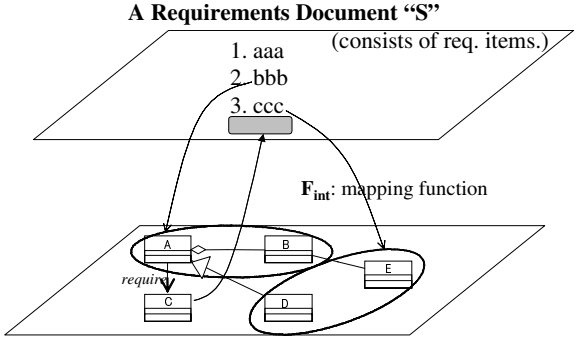


Figure 9: Mapping from Requirements to a Conceptual Graph

6 Related Work

In the area of requirements analysis and software specification, some studies to extract requirements models by applying NLP techniques to natural-language documents exist [9]. In particular, many of them derive OO models, e.g. class diagrams [10, 11, 12, 13] for software systems. Their techniques are basically to focus on nouns and verbs that are indicators of classes and of operations or relationships respectively, and their success greatly depends on the quality of an input document. For example, if mandatory descriptions are lacking from the document, the corresponding part of the model cannot be extracted. Since our approach uses multiple documents as inputs, our approach can mitigate these shortcomings. Furthermore they did not consider the extracted models as reusable assets like feature models. And, since we have adopted a variety of types of concepts and their relationships in our meta model of conceptual graphs so as to have wide applications for requirements analysis, we have developed a newly devised text-mining technique fit to our meta model in order to achieve the construction of the graphs from documents. In the area of database systems, a lot of work has also been done to derive a family of Entity Relationship (ER) models from natural-language documents and their major aims are designing a data schema [14, 15, 16, 17, 18]. They focused on the extraction of entities, attributes, relationships and inheritance ones, but did not consider the other constructs such as require relationships, which are necessary for requirements elicitation. Furthermore, an ER model can be derived from our conceptual graph in the same way as section 5.1, and in this sense, our resulting conceptual graph includes rich information for requirements modeling. CM builder, developed by Harmain et. al. [19], uses the domain knowledge, that has been made ready beforehand, to analyze semantically documents. More precisely, in their approach, the domain knowledge is extended to a more specific model by means of adding the extracted classes to it. Although our conceptual graph plays the same role on their domain knowledge of CM builder's technique, they did not discuss the technique how to construct the domain knowledge, i.e. conceptual graphs.

In research community of Ontology, many computerized tools for supporting ontology creation using text-mining techniques have been developed. Text2Onto of KAON [20, 21] is a computerized tool having a text-mining functions based on TF $\times$ IDF measure so that words frequently appearing can be extracted from text documents. In fact, our tool uses the same quantification techniques for word extraction. In [22], the author applied to software documents of a certain domain the technique similar to Text2Ont to extract the terminology that software developers, domain experts and other stakeholders could commonly use during software development processes. OntoLearn [23] adopted a kind of pattern matching technique to disambiguate words in the semantic analysis for word extraction. DODDLE [24] is also a tool to mining English texts for concept extraction based on term frequency, and it uses WordNet [25] and EDR dictionary [26] as a general-purpose ontologies. Although these tools developed by ontology communities have some functions to make our tool more elaborated, all of them cannot classify the extracted concepts and relationships into the types specific to requirements models as shown in Figure 2, e.g. "Class", "Function", etc. for concepts and "apply", "require", "perform", etc. for conceptual relationships. They are just for extracting concepts with no types and too general relationships such as "is-a", "has-a" and "synonym" etc. as general-purpose ontology or thesauruses, not necessarily suitable for requirements analysis. Their aim is different from ours and they are not immediate supports to requirements modeling. Our conceptual graphs have a variety of types of concepts and of relationships in order to apply to requirements modeling and elicitation, and these existing techniques could not classify the extracted concepts and relationships into these types. To support seamlessly requirements modeling, these techniques should extract not only concepts and their relationships but also their types that lead to the elements of requirements models.

As for the quality of input documents, [27] suggested several guidelines of writing natural-language sentences that could be used for extracting requirements models. Although they are for German, some of them could be useful to improve the quality of input documents for our tool.

7 Conclusion

In this paper, in order to support requirements modeling, we presented a computerized tool for extracting conceptual information from Japanese documents, and made several experiments to show the usefulness of our tool. Although our experiments mentioned in section 5 were too small in the sense of practical setting to argue the generality of the experimental findings, we could find the possibility of supporting the construction of useful conceptual graphs. According to the results of interviews to our subjects, the user interface of our tool should be improved.

None of conceptual graphs that our tool suggested included contradiction relationships, and our subjects added them by manual. The reason was that the documents we used did not contain any specific words denoting contra-

diction. We should explore more elaborated mining techniques together with good samples of documents.

Although our current approach is based on the frequency of words in documents, frequent words are not always important in general. Comparing different documents [28, 29] is one of the ways to complement this frequency based approach. Another way to create a conceptual graph of higher quality is the integration of many existing ontologies, including WordNet and EDR dictionary.

In sections 5.1 and 5.2, we illustrated how to derive two types of requirements models from our conceptual graphs. Formalization of these derivation rules using a graph rewriting system [30] and its automation are also a future work.

In section 5.3, we used our conceptual graph as domain knowledge. There are several excellent techniques and Meta CASE tools to generate domain specific modeling languages such as MetaEdit+ [31] , Metaview [32] and GME [33]. Our conceptual graph can be an input to these Mata CASE tools to produce domain specific modeling environments. This is one of the most interesting applications of our technique.

References

[1] K. Kang, S. Cohen, J. Hess, W. Novak, and A. Peterson. Feature Oriented Domain Analysis (FODA): Feasiblity Study. Technical Report CMU/SEI-90-TR-21, 1990.

[2] OMG. Unified Modeling Language Specification, Version 1.4. http://www.omg.org/cgi-bin/doc?formal/01-09-67.

[3] F. Harmsen. *Situational Method Engineering*. Moret Ernst & Young Management Consultants, 1997.

[4] M. Saeki. A Meta-Model for Method Integration. *Information and Software Technology*, 39:925 – 932, 1998.

[5] T. Tokunaga. *Information Retrieval and Natural Language Processing (in Japanese)*. University of Tokyo Press, 1999.

[6] Haruhiko Kaiya, Daisuke Shinbara, Jinichi Kawano, and Motoshi Saeki. Improving the detection of requirements discordances among stakeholders. *Requirements Engineering*, 10(4):289 – 303, Dec. 2005.

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

[8] M. Kitamura, R. Hasegawa, H. Kaiya, and M. Saeki. An Integrated Tool for Supporting Ontology Driven Requirements Elicitation. In *Proc. of 2nd International Conference on Software and Data Technologies (ICSOFT 2007)*, pages 73–80, 2007.

[9] 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.

[10] R. Abbott. Program Design by Informal English Descriptions. *Commun. ACM*, 26(11):882–894, 1983.

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

[12] S. Overmyer, B. Lavoie, and O. Rambow. Conceptual Modeling through Linguistic Analysis Using LIDA. In *Proc. of 23rd International Conference on Software Engineering (ICSE'01)*, pages 401–410, 2001.

[13] A. Montes, H. Pacheco, H. Estrada, and O. Pastor. Conceptual Model Generation from Requirements Model: A Natural Language Processing Approach. In *Lecture Notes in Computer Science (NLDB 2008)*, volume 5039, pages 325–326, 2008.

[14] R. Hausser. Database Semantics for Natural Language. *Artificial Intelligence*, 130(1), 2001.

[15] A. Min Tjoa and L. Berger. Transformation of Requirement Specifications Expressed in Natural Language into an EER Model, 1994.

[16] P. Chen. English Sentence Structure and Entity-Relationship Diagrams. *Information Science*, 29(2-3):127–149, 1983.

[17] S. Hartmann and S. Link. English Sentence Structures and EER Modeling. In *Proc. of 4th Asia-Pacific Conference on Conceptual Modelling (APCCM2007)*, pages 27–35, 2007.

[18] E. Buchholz, H. Cyriaks, A. Dusterhoft, H. Mehlan, and B. Thalheim. Acquiring Complex Information from Natural Language for EER Database Design. In *1st International Workshop on Applications of Natural Language to Data Bases (NLDB'95)*, 1995.

[19] H. Harmain and R. Gaizauskas. CM-Builder: An Automated NL-based CASE Tool. In *Proc. of 15th IEEE International Conference on Automated Software Engineering (ASE'00)*, pages 45–53, 2000.

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

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

[22] 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.

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

[24] T. Morita, N. Fukuta, N. Izumi, and T. Yamaguchi. DODDLE-OWL: A Domain Ontology Construction Tool with OWL. In *Lecture Notes on Computer Science (ASWC2006)*, volume 4185, pages 537–551, 2006.

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

[26] Japan Electronic Dictionary Research Institute. EDR Home Page. http://www.jsa.co.jp/EDR/index.html?

[27] G. Fliedl, C. Kop, W. Mayerthaler, H. Mayr, and C. Winkler. Guidelines for NL-Based Requirements Specifications in NIBA. In *Lecture Notes in Computer Science (NLDB 2000)*, volume 1959, pages 251–264, 2000.

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

[29] Akira Osada, Daigo Ozawa, Haruhiko Kaiya, and Kenji Kaijiri. Modeling Software Characteristics and Their Correlations in A Specific Domain by Comparing Existing Similar Systems. In Kai-Yuan Cai, Atsushi Ohnishi, and M. F. Lau, editors, *QSIC 2005, Proceedings of The 5th International Conference on Quality Software*, pages 215–222, Melbourne, Australia, Sep. 2005. IEEE Computer Society.

[30] G. Taentzer, O. Runge, B. Melamed, M. Rudorf, T. Schultzke, and S. Gruner. AGG : The Attributed Graph Grammar System. *http://tfs.cs.tu-berlin.de/agg/*, 2001.

[31] S. Kelly, K. Lyytinen, and M. Rossi. MetaEdit+ : A Fully Configurable Multi-User and Multi-Tool CASE and CAME Environment. In *Lecture Notes in Computer Science (CAiSE'96)*, volume 1080, pages 1–21, 1996.

[32] P. Sorenson, J. Tremblay, and A. McAllister. The Metaview System for Many Specification Environments. *IEEE Software*, 2(5):30–38, 1988.

[33] A. Ledeczi, M. Maroti, A. Bakay, G. Karsai, J. Garrett, C. Thomason, G. Nordstrom, J. Sprinkle, and P. Volgyesi. The Generic Modeling Environment. In *Proc. of WISP'2001*, 2001.