GOORE : Goal-Oriented and Ontology Driven Requirements Elicitation Method

Masayuki Shibaoka[1], Haruhiko Kaiya[2], and Motoshi Saeki[1]

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

Abstract. Goal oriented modeling methods are one of the promising approaches to elicit requirements. However, the difficulties in goal decomposition during requirements elicitation processes and a lack of the technique to utilize domain knowledge are obstacles to make them widely used in industry community. This paper proposes a method call GOORE where a domain ontology is utilized to support goal decomposition as domain knowledge to support goal decomposition. Although the experimental case study of feed readers suggested some points necessary to be improved, our approach can make goal oriented approach more powerful and more widely used.

1 Introduction

Goal-oriented modeling is one of the promising methodologies for requirements elicitation. In this methodology, customers' needs are modeled as goals, and the goals are decomposed and refined into a set of more concrete sub-goals. After finishing goal-oriented requirements analysis, the analyst obtains an acyclic (cycle-free) directed graph called goal graph. I There are several excellent studies on goal oriented methods such as I* [7], Tropos [2], KAOS [9], AGORA [4] etc. and their results are being put into practice. However, almost of them focused on languages including graphical languages and supporting tools. One of the obstacles to be spread to industries is a lack of powerful support of goal decomposition and refinement processes, i.e. supports for creating sub-goals of high quality. In these processes, stakeholders performing goal oriented analysis need high level of knowledge. For example, in early requirements elicitation phases for modeling business processes whether they are as-is or to-be [10], knowledge of a problem domain such as banking business is necessary, while the analysts should have much knowledge on the domain of software technology and an application such as Web computing and cryptography algorithms. Utilizing this kind of knowledge effectively allows us to perform goal decomposition and refinement processes of high quality.

This paper proposes two points; one is the technique to represent knowledge of a specific domain (domain knowledge) as an ontology and another is to embed the ontological system with goal-oriented analysis processes. We consider that an ontology consists of a set of concepts represented with a thesaurus and inference rules on them. By using inference rules, the sub-goals are deduced and suggested to the analysts. The

J.-L. Hainaut et al. (Eds.): ER Workshops 2007, LNCS 4802, pp. 225–234, 2007.

rest of the paper is organized as follows. The next section presents the basic idea and the structure of our ontologies. In sections 3, we show our method GOORE (Goal Oriented and Ontology driven Requirements Elicitation) and its supporting tool. Section 4 presents a case study of developing a feed reader to assess our approach.

2 Basic Idea

2.1 Goal Graph

Goal oriented methods are for supporting hierarchical decomposition and refinement of customers' requirements as the goals to be achieved. Basically, the resulting artifact after performing this method is an AND-OR graph whose nodes are elicited goals. Each goal is decomposed and refined into more concrete goals, sub-goals during the process. We have two types of goal decomposition; one is AND decomposition and another is OR. In AND decomposition, if all of the sub-goals are achieved, their parent goal can be achieved or satisfied. On the other hand, in OR decomposition, the achievement of at least one sub-goal leads to the achievement of its parent goal. Figure 1 illustrates a part of the goal graph of a software music player which can play the sound files of MP3, WPA etc, like Windows Media Player. The edges attached with an arc outgoing from a parent node show an AND decomposition, and for example, all of three goals "Control Playing", "Have Visualization Effects" and "Lightweight Processing" should be achieved in order to achieve their parent goal "Music Player" in the figure. On the other hand, either "Use Skins" or "Visualize Music", or both are necessary for the achievement of "Have Visualization Effects".

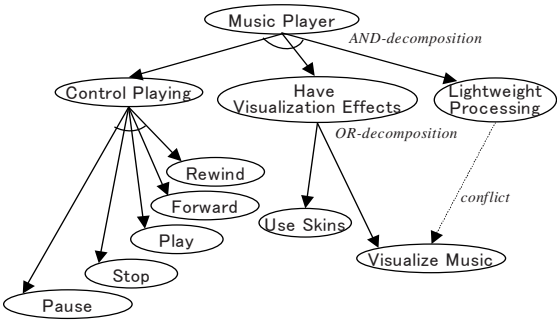


Fig. 1. An Example of a Goal Graph for Music Players

2.2 Using a Domain Ontology

As discussed in [6], 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. Each concept of a domain ontology can be considered as a semantic atomic element that anyone can have the unique meaning in a problem domain. The content of each goal is written in natural language and we use semantic processing of the goal content using a domain ontology.

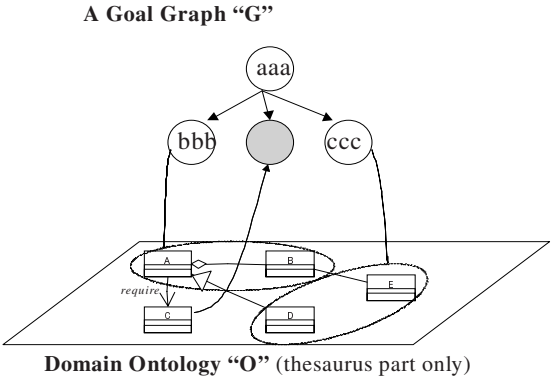
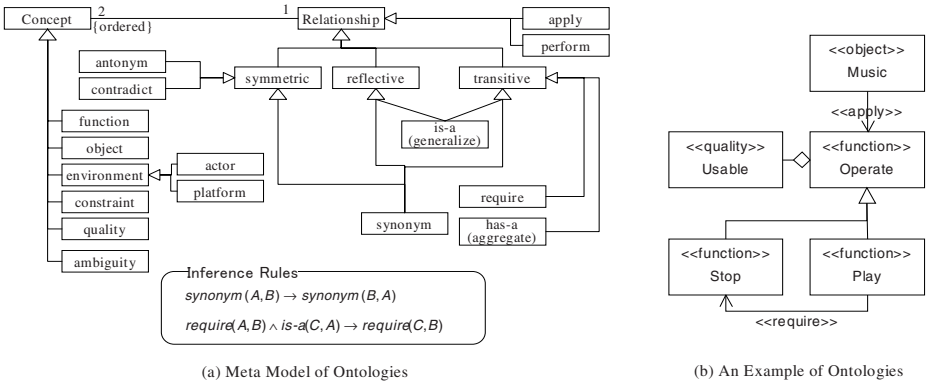


Fig. 2. Mapping from a Goal Graph to an Ontology

The words appearing in a goal description are mapped into words of the thesaurus part of the ontology, as shown in Figure 2. In the figure, the ontology is written in the form of class diagrams. Suppose that the goal description "bbb" contains the concepts A and B or their synonyms. The goal of "bbb" is mapped into the words A and B of the ontology, as shown in the figure. The logical inference on the ontology suggests to the analyst how to evolve his or her goal graph G. In the figure, although the graph G includes the goal node "bbb" mapped into the concept A, it has none of the goals mapped into 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 goal having C should be added to the graph G, say as a sub-goal shown in a grayed goal of the figure. In our technique, it is important what kind of relationship like "required by" should be included in a domain ontology for inference, and we will discuss this issue in the next sub section.

2.3 Domain Ontology

Figure 3 (a) shows the overview of a meta model of the thesaurus part of our ontologies. Thesauruses consist of concepts and relationships among the 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 3 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. The figure also includes a part of the inference rules. For example, the first and second rules express the reflectivity of synonym relationship and the inheritance of require relationship to a sub concept, respectively. Figure 3 (b) illustrates a part of the thesaurus for the domain of music players.



(a) Meta Model of Ontologies (b) An Example of Ontologies

Fig. 3. Ontology Meta model

3 GOORE Method and Supporting Tool

3.1 Elicitation Process

We call our method GOORE (Goal-Oriented and Ontology Driven Requirements Elicitation) and Figure 4 sketches the process for eliciting requirements following GOORE. The tasks to be preformed during the GOORE process can be divided into two cate-

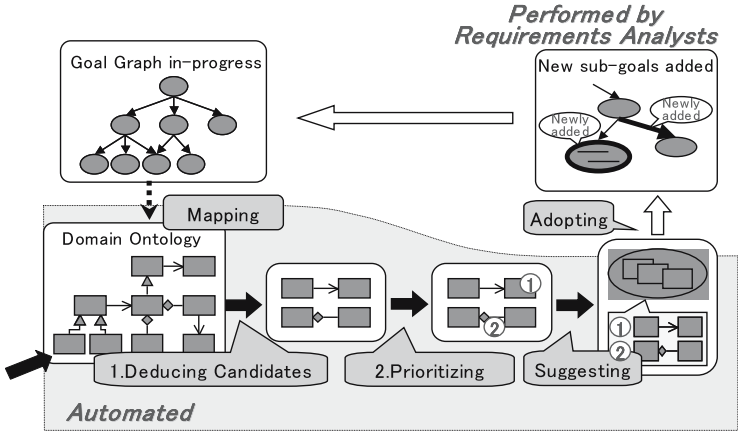


Fig. 4. GOORE Process

gories; one is a set of the activities that human requirements analysts should perform and another is the task that can be automated. The lower part, gray-shaded area of the figure expresses the automated tasks. The inputs of the automated task are a domain ontology and a goal graph in-progress that will be developed and evolved further. In the first step of the automated task, the natural language sentences of a goal description are

semantically analyzed by using a morphological analyzer[1]. The morphological analyzer also tells the part of speech of a morpheme (lexical categories such as nouns, verbs, adjectives, etc.), several morphemes that are not so significant can be removed from the sentences. For example, articles such as "a" or "the" are removed in general. After filtering out morphemes and identifying words and their parts of speech, the tool finds corresponding ontological concepts to each morpheme using a synonym dictionary. Thus mapping from the sentences into thesaurus part of the ontology is semi-automatically constructed. In the example of Figure 5 (a), the sentence of a goal includes the word "play" and it can be mapped into the ontological concept "play".

After constructing mappings, the inference mechanism on the ontology deduces and detects a set of the ontological concepts that should be added to the goal graph. The inference rules are described as a Prolog program. In Figure 5, the current version of the goal graph does not include a goal related to "stop", even though there is a require-relationship between Play and Stop in the ontology. Thus, the goal presenting Stop operation should be added to the goal graph shown in the left part of the figure. In addition, since the generalized concept Operate of Play has a quality concept Usable, the goal presenting the quality "Usable" can be suggested to add to the goal graph. Our inference rules are used for tracing the ontological relationships between the concepts and for detecting ones missing out of the goal descriptions. On account of space, the details of the rules are omitted and for the readers who have an interest to them, refer to [3].

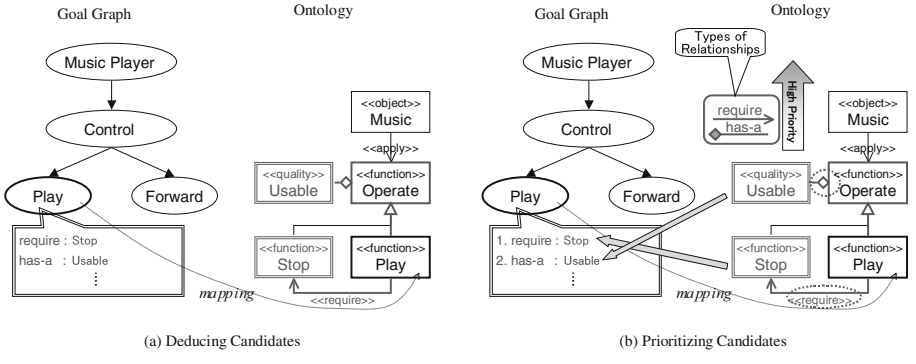


Fig. 5. Deducing and Prioritizing Candidates

Until the above steps, we can get several candidates to be added to the goal graph, e.g. Stop and Usable. Generally, there can be much more candidates suggested to the requirements analysts, as the goal graph becomes larger. The next step is prioritization. To avoid the explosion of candidates suggested, we attach priority to the deduced candidates and the candidates having high priority are selectively suggested. The priority can be calculated from numerical degrees attached to ontological elements, the structure of the goal graph being constructed, the inference process and records of the

[1] Since our supporting tool is currently Japanese version, we use Japanese morphological analyzer called SEN (https://sen.dev.java.net) written in Java.

analyst's selection activities. This calculation mechanism is embedded into the Prolog program and during the inference on the ontology, a candidate and its priority are simultaneously calculated. In the example of Figure 5 (b), consider that two candidates "Stop" and "Usable" are deduced from "Play". We attach to relationship type "require" higher priority degree rather than "has (aggregation)". Furthermore the relationship "is (generalization)" is used to deduce the "Usable". That is to say, to reach "Usable", two relationships "is" and "has" should be traced, while only one relationship is used for getting "Stop". The case where the fewer relationships are used for getting deduction results has the higher priority. Thus, the candidate "Play" is listed with higher priority than "Usable", as shown in Figure 5 (b). According to the ontological relationships and concepts, we provide constant numerical values as priority degrees, e.g. "require" has 5 and "has" has 3 etc. And the priority degree is also attached to each inference rule, and whenever the rule is used for deducing a candidate, its priority degree is numerically added. The number of inference steps is a factor to calculate the priority and the fewer step gives the higher priority value. The priority value is dynamically calculated and changed. Suppose that the analyst does not select the highest priority "Stop" in Figure 5 (b). At the next calculation, the priority degree of the candidate "Stop" is reduced. One of the benefits of using Prolog is flexibility of calculation rules of priority degrees.

3.2 Using a Supporting Tool

We have developed a prototype of the supporting tool for GOORE method. It architecturally consists of two parts; an editor of goal graphs and an inference engine to deduce and prioritize candidates. The former part was implemented as Java program, while in the latter Prolog was used. Thus, our ontologies are defined as facts of a Prolog program. In the example of Figure 3, the concepts Play and Stop and their require-relationship can be represented as a Prolog fact `require(function(Play), function(Stop))`. Each ontological concept is defined using a Prolog functor labeled with its type and the concept, e.g. `function(Play)` and `quality(Usable)`, while a relationship corresponds to a binary predicate like `require(x,y)` and `has(z,w)`.

The tool has the functions; 1) inputting and editing goal graphs in graphical form, i.e. a graphical editor for goal graphs, 2) getting an ontology described in Prolog, 3) deducing candidates and prioritizing them as background tasks of Prolog execution, and 4) showing a list of the candidates following their priorities to an analyst as a menu so as to make him or her select suitable ones out of them. The tasks of deducing, prioritizing and showing the candidates are done in real-time during the analyst's activities of inputting and editing the goal graph.

Figure 6 illustrates a screenshot of the tool. In the figure, an analyst is constructing a goal graph of a feed reader (news aggregator), which is a client software to retrieve frequently updated Web contents of certain topics from blogs and mass media web sites etc. She uses the ontology of Feed Reader and is refining the goal "User Registration" (for the feed reader). The candidates of newly added goals related to "User Registration" are shown in a small window. The mark ⊞ attached to a goal shows the existence of the deduced candidates, and she can get the candidate list if she clicks this mark. Figure 7 shows the process of selecting a candidate as a sub-goal of "User Registration".

As shown in the left screen, the analyst selects as a sub-goal the candidate “Setting a Personal Profile”, which has the second priority in the candidate list, and gets a new goal graph shown in the right screen of the figure.

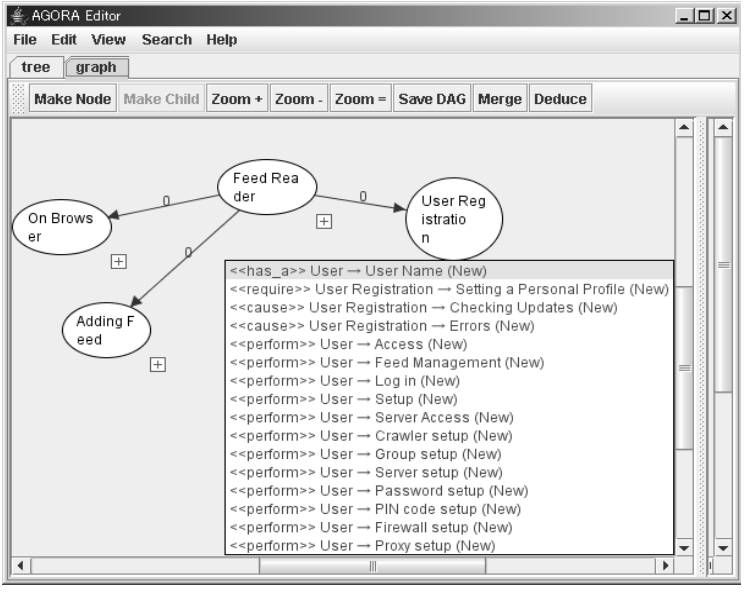


Fig. 6. Screen of the Supporting Tool

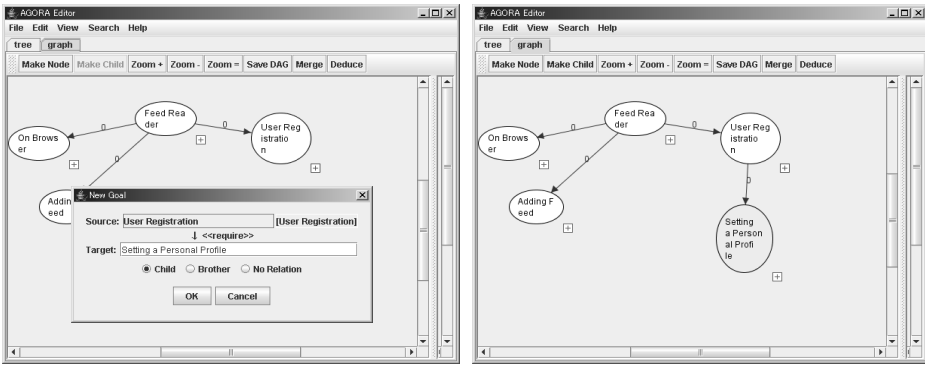


Fig. 7. Selecting a Candidate on the Supporting Tool

4 Case Study

We had a small experimental case study to investigate the usefulness of our approach. We used the example of Feed Reader, which was mentioned in the previous section, and asked our subjects to construct its goal graph. Before starting this experiment, we

developed a domain ontology of Feed Reader. It consists of 593 concepts and 1023 relationships as a thesaurus part, and has 8 types of inference rules and 10 types of calculation rules for prioritizing candidates. Our supporting tool is set to suggest 16 candidates at a time for each goal.

Figure 8 illustrates the process of our experimental case study. One of us, the expert of Feed Reader, completed its requirements specification in the form of itemized sentences as a right answer of this example problem. This specification included 93 requirements items. In addition to this specification, he composed an initial requirements list having 10 items and provided it for our subjects as a customer's initial requirements. We had four subjects, two of them did not have sufficient knowledge of feed readers, while two of them used our GOORE method and the supporting tool. After they got the initial requirements list, they constructed goal graphs in an hour and then extracted requirements specifications from their constructed graphs. We recorded their activities including tool manipulations with a video camera and a screen recorder. They could communicate with one of us, who played a role of customers, in order to resolve their unclear points. After they finished these construction tasks, we had interviews and questionnaires to evaluate usability of our method and tool. In addition, to check the quality of resulting goal graphs, we compared them with the right answer that the expert developed before.

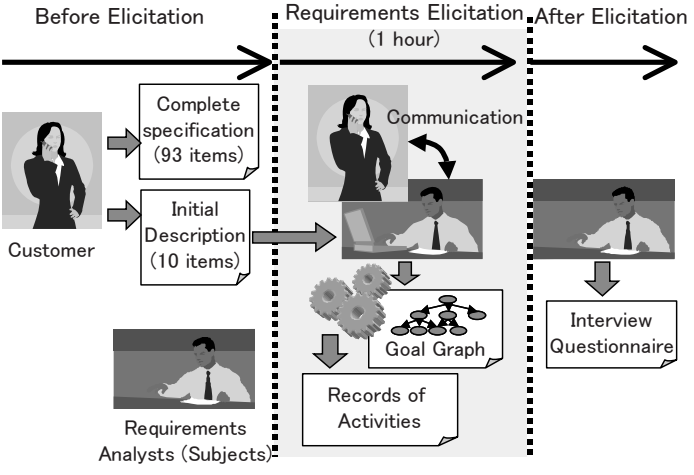


Fig. 8. Experiment

Table 1 summarizes the experimental result. Each column from the second expresses the result of a subject. In the example of the third column, the subject (No, Yes) means that the subject had no knowledge of feed readers and used our supporting tool. He got 43 goals, 10 of which resulted from the candidates suggested by the tool. He selected the candidate of rank average 5.9 out of 16 suggested candidates, and it shows that our adopted priority calculation would be reasonable. Furthermore 40 requirements items could be extracted from the 43 goals, and 34 of them could be considered as right answers because these 34 were also included in the complete specification. 9 of the 10

Table 1. Experimental Result

Evaluation Items	Result of a Subject†			
	(No, No)	(No, Yes)	(Yes, No)	(Yes, Yes)
#Goals	42	43	63	48
#Goals obtained by Suggesting Candidates	—	10	—	13
#Goal Relationships by Suggesting Candidates	—	0	—	0
Average of Ranks of Adopted Goals (in 16 items)	—	5.9	—	2.4
# Requirements Items included in the Goal Graph‡	38	40 (10)	71	60 (11)
# Requirements Items included in the Goal Graph and the Completed Artifacts‡	32	34 (9)	56	55 (10)
Expiration Time for Constructing a Goal	70sec.	76sec.	55sec.	75sec.
Longest Time for Deducing Candidates	—	23sec.	—	68sec.

† (having domain knowledge or not, using a GOORE method & a tool or not)

‡ the number in ( ) stands for the requirements items included in the adopted candidates

adopted candidates were also included in the completed specification, and it shows that the suggested candidates had high quality. In the case of another subject using the tool, i.e. (Yes, Yes), the reasonability of prioritization (rank 2.4) and the high quality of the suggested candidates were clearer.

Negative results are that 1) the tool users did not find goals much more than non-users and 2) the tool users spent longer time for constructing a goal.

5 Conclusion and Future Work

In this paper, we presented an extended version of a goal oriented modeling method which was combined with an ontological technique to utilize domain knowledge. Furthermore we developed a supporting tool following the proposed method and had a experimental case study. From the results of the case study, although our approach has still some deficiencies, we found that we should keep our approach, GOORE. The future research agenda can be listed as follows.

1. Combining GOORE with other methods, in particular scenario based approach
   Since our approach uses natural-language description, it would be more effective in the cases when more natural-language sentences are described such as scenario oriented methods [8].
2. Improving how to suggest deduced candidates to analysts
   The current version of the tool shows the candidates in the form of words and phases only. In our case study, the subjects could not understand the correct meanings of the suggested words and phases, and were sometime confused to hold precise intents of the suggestions. This shortcoming caused longer time consumption for the tool users to construct their goal graphs in our experiment. Ontologies can include this kind of information.

3. Constructing ontologies of high quality
   Constructing an ontology by hand is time-consuming. Currently, we consider the application of text mining techniques to extract ontological elements from natural-language documents in a domain [1,5].

References

1. KAON Tool Suite, http://kaon.semanticweb.org/
2. Castro, J., Kolp, M., Mylopoulos, J.: A Requirements-Driven Development Methodology. In: Dittrich, K.R., Geppert, A., Norrie, M.C. (eds.) CAiSE 2001. LNCS, vol. 2068, pp. 108–123. Springer, Heidelberg (2001)
3. Kaiya, H., Saeki, M.: Using domain ontology as domain knowledge for requirements elicitation. In: RE 2006. Proc. of 14th IEEE International Requirements Engineering Conference, pp. 189–198. IEEE Computer Society Press, Los Alamitos (2006)
4. Kaiya, H., Horai, H., Saeki, M.: AGORA: Attributed Goal-Oriented Requirements Analysis Method. In: RE 2002. IEEE Joint International Requirements Engineering Conference, pp. 13–22. IEEE Computer Society Press, Los Alamitos (2002)
5. Kitamura, M., Hasegawa, R., Kaiya, H., Saeki, M.: An Integrated Tool for Supporting Ontology Driven Requirements Elicitation. In: ICSOFT 2007. Proc. of 2nd International Conference on Software and Data Technologies (2007)
6. Maedche, A.: Ontology Learning for the Semantic Web. Kluwer Academic Publishers, Boston (2002)
7. Mylopoulos, J., Chung, L., Yu, E.: From Object-Oriented to Goal-Oriented Requirements Analysis. Communications of the ACM 42(1), 31–37 (1999)
8. Rolland, C., Souveyet, C., Achour, C.B.: Guiding Goal Modeling Using Scenarios. IEEE Transaction on Software Engineering 24(12), 1055–1071 (1998)
9. van Lamsweerde, A.: Goal-Oriented Requirements Engineering: A Guided Tour. In: RE 2001, pp. 249–263 (August 2001)
10. Yu, E.: Towards Modeling and Reasoning Support for Early-Phase Requirements Engineering. In: RE 1997. Proc. of 3rd IEEE International Symposium on Requirements Engineering, pp. 226–235. IEEE Computer Society Press, Los Alamitos (1997)