Asian Journal of Information Technology 5 (9): 985-989, 2006

Design and Development of Ontology for Natural Language Requirement Specification by Eliciting Object Oriented Elements

[1]G.S. Anandha Mala and [2]G.V. Uma
Department of Computer Science and Engineering, College of Engineering,
Anna University, Guindy, Chennai, Tamil Nadu, India-600025 India

Abstract: Ontologies are especially useful for the development of high-level reusable software, like domain models and frameworks. They provide an unambiguous terminology that can be shared by all involved in the development process. Developing ontology for the natural language requirement specification by eliciting object oriented elements namely classes, attributes, methods, relationships between the classes by applying natural language processing techniques is proposed in this study, which knowledge engineers can use for requirement analysis. The elicitation and modelling of software requirements are accomplished in three steps. As a first step, domain knowledge (object oriented elements) is elicited by subjecting the problem statement written in natural language (English) to object oriented analysis. From the acquired domain knowledge, resource description framework has been generated and visualized as ontology with the help of the tool GATE 3.0 (General Architecture for Text Engineering) then. Finally the ontology has been stored effectively into the database, which ease the task of querying and acquiring the domain knowledge. This way, the same ontology can be used to guide the development of several applications, diluting the costs of the initial stage and allowing knowledge sharing and reuse.

Key words: Natural language processing, object oriented analysis, object oriented design models, ontology

INTRODUCTION

The ontology is a representation vocabulary often specialized to one domain or subject matter. Domain ontologies are formal description of the classes of concepts and the relationships between those concepts that describe an application domain. The design and development of ontology for software requirement specification tries to bring out the domain information in the ontological form from the requirement specification to develop an application. The ontology can be reused for developing any other application on the same domain[1].

Recent study in artificial intelligence is exploring the use of formal ontologies as a way of specifying content specific agreements for the sharing and reuse of knowledge among software entities. The software industry needs to maintain the domain information and they have to be shared and reused in the industry among the developers during the development of the application, to make domain assumptions explicit and also to separate domain knowledge from operational knowledge. This proved to be the need behind creating and developing the ontology.

Related works: Although it has been proven that Natural Language processing with holistic objectives is very complex, it is possible to extract sufficient meaning from NL sentences to produce reliable models. Advances in the field of requirements elicitation is discussed in this section, which is followed by the various definitions given for ontology.

The first relevant published technique attempting to produce a systematic procedure to produce design models from NL requirements was Abbot[2]. Abbot suggested a non-automatic methodology that only produces static analysis and design products obtained by an informal technique requiring high participation with that of users for decisions. Methods to bring out a justified relationship between the natural- language structures and OO concept are proposed by Sylvain[3] who show that computational linguistic tools are appropriate for preliminary computer assisted OO analysis. Sawyer in their REVERE[4] makes use of a lexicon to disambiguate the word senses thus obtaining a summary of requirements from a natural language text but do not attempt to model the system. Liwu Li[5] also presents a semi-automatic approach to translate a use case to a sequence diagram. It needs to normalize a use case manually. Overmyer[6], also present only a complete interactive methodology and prototype. However, the text analysis remains in good part a manual process. Liu[7] present an study, which uses formalized use cases to

Corresponding Author: G.S. Anandha Mala, Department of Computer Science and Engineering, College of Engineering, Anna University, Guindy, Chennai, Tamil Nadu, India-600025 India

capture and record requirements. Ke Li[8] also semi-automate the process of requirement elicitation where the text is matched with predefined statements. If there is no match then get help from user to clarify incomplete/ambiguous data. Participation of domain experts, customer are needed in class identification process in contrast to our fully automatic methodology which is named here as Domain Knowledge Elicitor.

This study aims at providing the facilities of sharing and reuse of the domain information among the developers in the form of ontology. It is possible to find in the literature several definitions for ontology. One of the most cited is the one proposed by Gruber, "An ontology is a formal, explicit specification of a shared conceptualisation[9]. The definition proposed by Gruber is general; however, ontology can be defined in specific contexts. For example, taking the paradigm of agents into account[10] establish that ontology is a formal description of the concepts and relations, which can exist in a community of agents. The importance of the terms of ontology can be perceived in the next definition: "An ontology is a hierarchically structured set of terms to describe a domain that can be used as a skeletal foundation for a knowledge base[11].

More recent definitions of ontologies are the following ones: "An ontology is a common, shared and formal description of important concepts in an specific domain[12]; An ontology is a formal explicit representation of concepts in a domain, properties of each concept describes characteristics and attributes of the concept known as slots and constrains on these slots[13]. Sometimes concepts are termed classes, properties are also known as roles while facets are used rather than slots. An ontology is a theory which uses a specific vocabulary to describe entities, classes, properties and related functions with certain point of view[14]. An ontology necessarily includes a specification of the terms used, (terminology) and agreements to determine the meaning of these terms, along with the relationships between them[15].

Like Fonseca's[14] definition of ontology, the method proposed in this study for the construction of ontology consists of classes, subclasses, attributes, methods and the relationships among the classes. The ontology can be constructed for any domain with the help of Requirement Specification. The pre-processor module of Domain Knowledge Elicitor in the system architecture shown in Fig. 1. identifies the key concepts required for the construction of ontology.

Proposed system architecture: Ontologies are key elements required to enable knowledge exploitation and information retrieval systems. Essentially, domain ontologies are made of sets of concepts and the relationships that can be expressed among those concepts. Since the building blocks of domain ontologies are concepts and relations among concepts that describe an application domain, NLP techniques are used to retrieve the object-oriented concepts namely classes, attributes, methods and relationship among the classes. The concepts identified are represented in the form of ontology and effectively stored in the database for easy information retrieval. The method of visualizing the requirements specification as ontology by applying natural language techniques is explained here. The proposed system architecture is shown in Fig. 1. The system has the following modules, 1) Domain Knowledge Elicitor, 2) RDF Generator 3) RDF Parse Engine 4) Data Storage and Query Engine

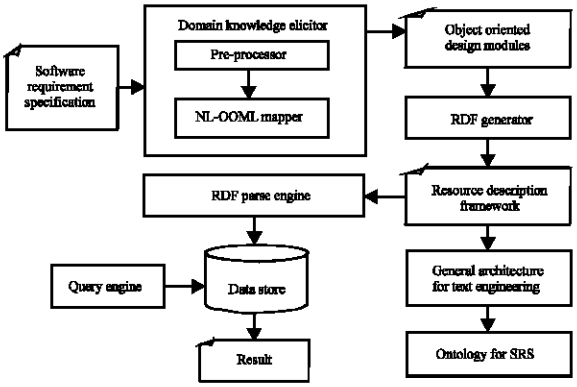

Fig. 1: The system architecture

Domain knowledge elicitor: In all the earlier study mentioned about requirements elicitation, the process is not fully automatic. User assistance is required in pronoun resolution. The proposed methodology for domain knowledge elicitation includes the automatic reference resolution, which eliminates the user intervention as in the previous study. It takes an input a natural language text describing the requirements for a system and identifies the object-oriented system elements namely classes, attributes, methods and relationships among the classes. Domain model elicitor does this job with the help of pre-processor and NL-OOML mapper.

Pre-processor: The given input problem statement is split into sentences to ease the further processing. Each sentence of the input text is subjected to tagging in order to get the parts of speech marker for every word in a sentence. Tagging of the words is necessary to chunk the words that form a noun or verb phrase. Also the words that are candidates for classes, attributes, methods, use cases and actors have to be chosen depending upon their tags. We make use of the Brill tagger for this purpose. The noun and the verb phrases are identified based on simple

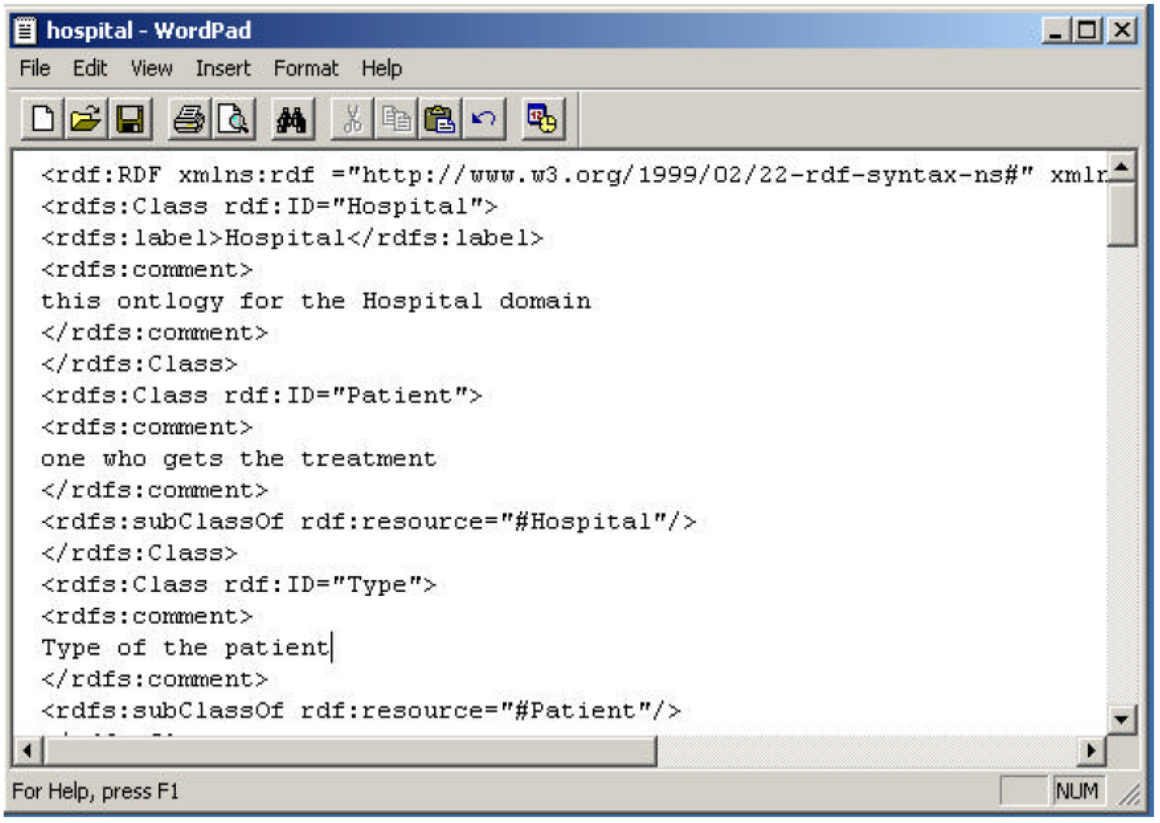

Fig. 2: Sample RDF Code

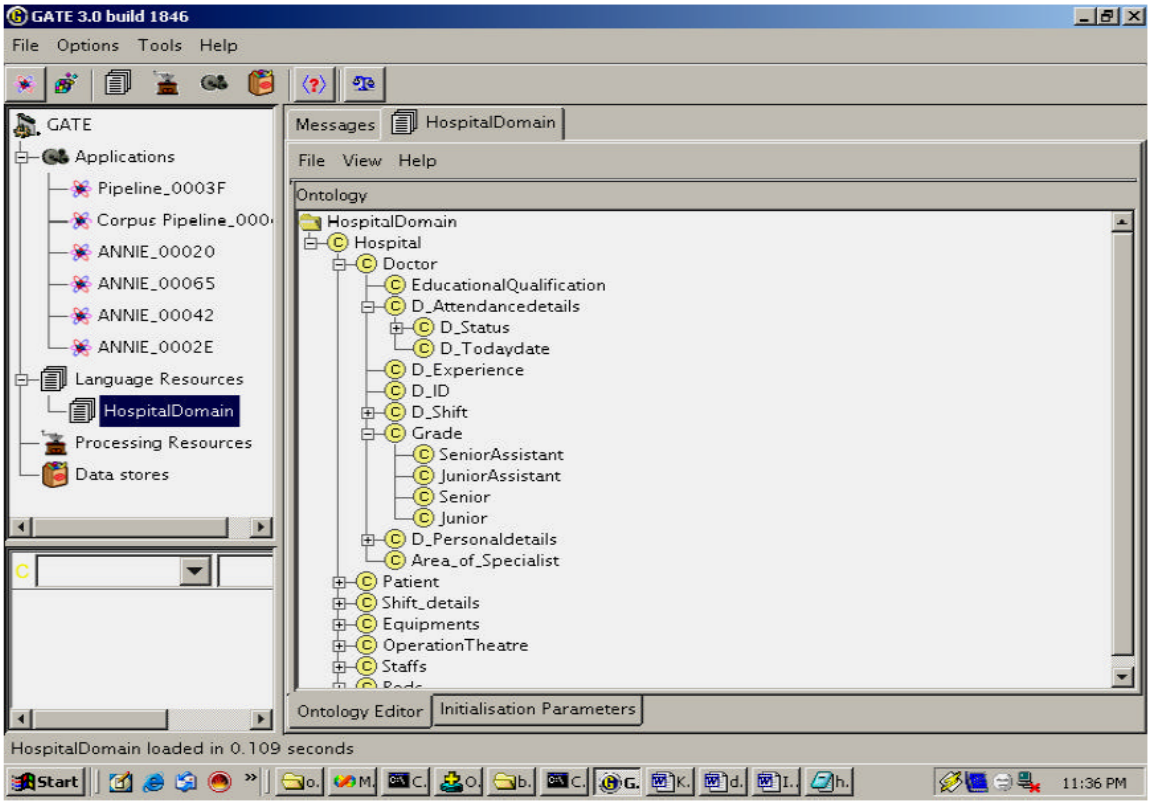

Fig. 3: Ontology

phrasal grammars. In the sentence, the subjects and objects sometimes happen to be pronouns. In that case, they have to be resolved to their respective noun phrases. We make use of the Mitkov's 'Pronoun resolution with limited knowledge' algorithm for this task and get all pronouns resolved.

NL-OOML mapper: The NL-OOML mapper accepts a preprocessed problem description as input. It recognizes nouns, verbs, adjectives and prepositions in a sentence from the preprocessor. Simple rule based study is followed for identifying the object oriented elements. List of rules are given below:

Rules to generate class diagram constituents:

- Translating Nouns to Classes. A noun, which does not have any attributes, need not be proposed as a class.
- Translating Noun-Noun to Class-Property according to position. When two nouns appear in sequence in the text, the first Noun is translated to Class and the following Noun is translated to properties of this Class.
- A simple heuristic is used to decide which nouns are classes and which form the attribute. In Noun-Noun, if the first noun is already been chosen as the class then the second noun is taken as the attribute. The attributes are decided based on the verb phrase.
- Translating the lexical Verb of a non-personal noun to a Method of this noun.
- Matching a Noun to a Personal Pronoun as the nouns of previous sentence. The pre-processor does this.

Rules for mapping to relationships: After identifying the nouns and verbs and creating a simplified canonical noun-verb-noun structure we are in a position to map this information onto classes and their relationships with each other.

Two key steps are:

- If the sentence uses the passive voice, convert S-Ved to V-O(S)
- Translating the lexical Verb to a Method of the classes formed from the two Nouns either side of this Verb, so that a relationship can be created between the classes.

RDF generator: After successful completion of domain knowledge extraction, the RDF Generator generates the RDF (Resource Description Framework) file using the domain knowledge elicited from the software requirement specification. The Generator concentrates on the class, subclass and properties of the RDF. The sample RDF code generated for the requirements specification of hospital management system is shown in Fig. 2.

The RDF generated is fed as input to the GATE tool to visualize the RDF as ontology. The ontology for the hospital management system is shown in Fig. 3.

RDF parse engine: The RDF generated is parsed to extract the information and stored in the dynamic database. The Parsing Algorithm is a generalized one, which takes care of the classes, subclasses and properties. The parsing algorithm of RDF is given below;

- The complete RDF file of the domain is read and stored in a string.
- The String Tokenizer is used for splitting the RDF file into tokens
- Finding out if there exist more tokens after tokenizing.
- Check the each token.
- If the token is rdfs :class then the rdf:ID is extracted by identifying the index position of the " and stored in the database.
- If the token is rdfs:subclassof then the resource is extracted by identifying the index position of # and stored in the database.
- If the token is a comment then the next token is extracted and stored in the database.
- The Process is done until the end of class is identified and all the extracted information is stored in the database.
- The Properties is extracted and the domain and the range value are stored in the database.

Data storage and querying: The database is designed effectively to store domain information extracted and maintain the data in three Tables. First table stores each class along with its subclasses and descriptions. The second one stores the properties along with its domain. The third one stores the range of the properties along with domain for which the properties belongs. The database can be queried to get the details like what kind of relationships holds between the classes and what can be the attribute value etc.,

RESULTS AND DISCUSSION

The entire system was implemented using JAVA and the 'Domain Knowledge Elicitor' was validated using 100 problem samples each of around 500 lines. The result produced by the system was compared with that of the human output. The human outputs were the results that were obtained by conducting the noun-verb analysis on the text. It was considered as the baseline and taken as expert judgment. The system does not miss to identify

any of the classes and methods. But approximately 12.4% of additional classes and 7.4% of additional methods are identified in the entire sample taken, those that are removed by human by intuition that they may not be classes. Since system lacks that knowledge, they are listed as classes. The missed out methods occur only if the tagger assigns a wrong tag to the word. Also the system perfectly identifies all the attributes and actors with out any additional, missed or miss assignments.

The acquired domain knowledge is visualized as ontology and stored in a database, which makes the retrieval of information quick and easy. We have tried to bring out a generic system in which the RDF is generated for all the SRS irrespective of the domain.

CONCLUSION

The study presents an approach to develop ontology for the natural language requirement specification text by acquiring domain knowledge. Natural language processing techniques and set of rules are used to elicit domain knowledge from the requirement specification. The deficiencies in the tagger and the reference resolver present in the preprocessor can be overcome by building a knowledge base which can also improve the effectiveness of generation of the system elements.

REFERENCES

1. María Auxilio Medina Nieto, 2003. An overview of ontologies. Technical report, center for research in information and automation technologies interactive and cooperative technologies Lab Universidad De Las Américas Puebla-México.
2. Abbot, R.J., 1983. Program design by informal English descriptions. Communications of the ACM, 26,pp: 882-894.
3. Sylvain Delisle, Ken Barker and Ismaïl Biskri, 1999. Object-Oriented Analysis: Getting help from robust computational linguistic tools, in G. Friedl, H.C. Mayr (Eds.) Application of Natural Language to Information Systems, Oesterreichische Computer Gesellschaft, pp: 167-172.
4. Sawyer, P., P. Rayson and R. Garside, 2002. REVERE: Support for requirements synthesis from documents. Information Systems Frontiers J., 4,pp: 343-353.
5. Liwu Li, 1999. A semi-automatic approach to translating use cases to sequence diagrams. Proceedings of technology of object-oriented languages and systems, July, IEEE CS Press, pp: 184-193.
6. Overmyer Scott, P. and B. Lavoie Rambow, 2001. Conceptual modelling through linguistic analysis using LIDA. Proceedings of the 23rd International Conference on Software Engineering, ICSE 2001, Toronto.
7. Dong Liu, Kalaivani Subramaniam, Behrouz H. Far and Armin Eberlein, 2003. Automating transition from use cases to class model. MSc Thesis, University of Calgary.
8. Ke Li, 2005. Towards semi-automation in requirements elicitation: Mapping natural language and object-oriented concepts 13th IEEE International Requirements Engineering Conference.
9. Gruber, T.R., 1993. A translation approach to portable ontology specifications. Knowledge Acquisition, 5,pp: 199-220.
10. Russell, S. and P. Norvig, 1995. Artificial intelligence: A modern approach. Prentice Hall, Englewood Cliffs, NJ.
11. Swartout, B., R. Patil, K. Knight and T. Russ, 1996. Toward distributed use of large-scale ontologies. Proceedings of the tenth knowledge acquisition for knowledge- based systems workshop, Banff, Alberta, Canada.
12. Fensel, D., I. Horrocks, F. Harmelen, S. Decker, M. Erdmann and M. Klein, 2000. OIL in a nutshell, Proceedings of the European knowledge acquisition Conference, (EKAW-2000), R. Dieng et al. (Eds), Lecture Notes in Artificial Intelligenc, LNAI, Springer-Verlag, October.
13. Noy, F.N. and D.L. McGuinness, 2001. Ontology Development 101: A Guide to creating your first ontology. Stanford knowledge systems laboratory technical report KSL-01-05 and stanford medical informatics technical report SMI-2001-0880.
14. Fonseca, F., M. Egenhofer, P. Agouris and G. Camara, 2002. Using ontologies for integrated geographic information systems. Transactions in GIS, pp: 3.
15. Starlab, 2003. Systems technology and applications research laboratory home page. Faculty of Sciences, Department of Computer Science, Vrije Universiteit Brussel. Available at: http://www.starlab.vub.ac.be/default.html