International Journal of Computer Information Systems and Industrial Management Applications.
ISSN 2150-7988 Volume 6 (2014) pp. 227 - 236

Domain Ontology Based Class Diagram Generation From Functional Requirements

Jyothilakshmi M S and Philip Samuel
Information Technology, School of Engineering
Cochin University of Science and Technology
Kochi-22 Kerala India
jyothi.shaji@gmail.com; philips@cusat.ac.in

Abstract **—Domain ontology formally represents knowledge as a set of concepts within a domain, and the relationships among those concepts. This paper proposes a method to generate class diagram from functional requirement specification which is written in natural language by using Domain ontology and Natural language processing techniques. Major steps in this method are identification of nouns and verbs from requirement specification statements and linking them with the ontology. From the ontology we get information about core concepts and relationships among those concepts. Thus domain ontology helps in the identification of classes, attributes and relationships for the particular domain for which the system is to be developed.**

***Keywords— Domain Ontology, Natural language processing, UML Class Diagram*.**

I. INTRODUCTION

Requirements are generally expressed in the form of natural language statements [1]. Analysis and Design Process covers two major phases of software development life cycle. The two phases are often overlapped due to the recursive nature of requirements. Process usually starts with getting know-how of customer business processes and existing systems, understanding customer needs, expectations, constraints and elicitation of requirements. Requirements conflicts are removed, and issues and concerns are addressed to develop a clear understanding of customer requirements. These requirements are then used to define the functionality of proposed system and to determine what the systems are intended to do. System functionality and architecture is documented, and after reviewing internally, communicated to customer for the purpose of validation. This document then gets mature as a result of customer feedback and serves as the basis for further development. For manual requirement analysis and design process, Software engineers spent lot of time.

A class diagram in the Unified Modeling Language (UML) is a type of static structure diagram that describes the structure of a system by showing the system's classes, their attributes, operations (or methods), and the relationships among the classes. It is the main building block of object oriented modeling and design. In the diagram, classes are represented with boxes which contain three parts. The upper part holds the name of the class, the middle part contains the attributes of the class and the bottom part gives the methods or operations the class can take or undertake. The relationship among the classes can be association, dependency, aggregation, generalization and composition. In the design of a system, a number of classes are identified and grouped together in a class diagram which helps to determine the static relations between those objects.

The aim of this paper is to automate the extraction of class diagrams from functional requirement specification which is represented in natural language statements and thus to reduce the problems associated with manual extraction.

In this approach we utilize the benefits of domain ontology. Ontology is a "formal explicit specification of a shared conceptualization" [2]. It formally represents knowledge as a set of concepts with in a domain and may be used to describe the domain. All the core concepts and their relationships are represented in the ontology in a formal way so that we can automate the process of understanding the domain specific terms. Formal languages used to construct ontology are known as ontology languages. Web Ontology Language (OWL) is one the most recent development in standard ontology language from World Wide Web Consortium (W3C). In our method ontology is provided in OWL format. We can identify the classes, attributes and inter class relationships by comparing with the terms in the ontology with that of functional requirement specification. Also ontology helps us to identify certain domain specific terms which are not explicitly mentioned in the functional requirements. When ontology cannot identify the concepts or terms in the natural language statements, we go for some natural language processing techniques to extract the classes, attributes and inter class relationships.

Our paper is organized as follows. Section 2 analyzes previous works on class diagram generation from natural language requirements. Section 3 is the description of proposed approach. Section 4 is the conclusion.

II. RELATED WORKS

Few approaches are already developed for the extraction of class diagram from natural language requirement specification. In this section we survey those works.

A. *Auto-generation of Class Diagram from Free-text Functional Specifications and Domain Ontology by Xiaohua Zhou and Nan Zhou*

Zhou and Zhou [3] propose a methodology that uses NLP and domain ontology. In this method some initial processing of the natural language requirements is done with the help of some linguistic patterns. Also in the domain ontology each concept is provided in some specific format as input and it is used to refine the result.

The approach applies a NLP-based spider model to search classes of interest, that is, they identify core classes of the domain as starting point, and further find classes that are related with the identified ones. The major components of the approach are described below.

Domain Ontology serves as domain knowledge to improve the performance of conceptual modeling.

Candidate Class Identification module outputs Preliminary Candidates and Refined Candidates, which serve as input of spider model. The part of speech (POS) tagger and sentence parser produce preliminary candidates. Further, word sense disambiguation technique and semantic network are employed to produce refined candidates, which are semantically related to the concepts defined in the ontology.

Relationship Identification module is the most important component of the spider model. It uses linkage distance to identify all concept pairs with strong semantic association within a sentence. Then assigns different weight to each concept pair to indicate how strong the connection is according to the constituents the concepts serve as in the sentence.

Attribute Identification module distinguishes attributes from classes. After two concepts are found to be strong associated to each other, then whether the concepts stand for a class or an attribute of a class is identified.

Naming Relationship module applies linguistic patterns to find aggregation relationship and generalization relationship. It uses numeric relationship understanding technique to distinguish one-to-one association p, one-to-many association, and many-to-many association.

Parallel Structure uses the feature of English sentence structure to help elicit more attributes of a class, more child classes of a super class, and more aggregation part of a class.

Throughout this method NLP techniques are applied. Part-of-speech (POS) tagger and sentence parser are used to produce preliminary class candidates while word sense disambiguation (WSD) and semantic network are employed to refine preliminary candidates. Linkage distance produced by Link Grammar Parser is used to find concept pairs within strong semantic association within a sentence. Different weights are assigned to each concept pair according to the linguistic patterns the concept pair belongs to. Linguistics patterns are used to evidence class-attribute relationship, aggregation relationship, and generalization relationship, and distinguish three types of association relationships. Parallel structure serves as a clue to find more attributes of a class, more child classes of a super class, and more parts of a composite class.

Secondly, this approach well integrates domain ontology with class generation for the first time. The domain ontology contains important or core domain knowledge. The ontology feeds the system important classes and their attributes. However this method needs a set of linguistic pattern as input.

B. Class diagram extraction from textual requirements using Natural language processing (NLP) techniques by Mohd Ibrahim, Rodina Ahmad

Mohd Ibrahim, Rodina Ahmad [4] proposes a methodology that also uses NLP and domain ontology. The workflow of this method is as follows. From the requirements document stop words are identified and stored in list. With the help of taggers sentences are tagged and thus nouns, noun phrases and verbs are identified. Then by applying RACE stemming algorithm the root form of each word is find out and stored. Now by applying Class Identification Rules, Attribute Identification Rules and Relationship Identification Rules the components are identified. The rules are given below.

1) Class Identification Rules:

C-Rule1: If a concept is occurred only one time in the document and its frequency is less than 2 %, then ignore as class.

C-Rule2: If a concept is related to the design elements then ignore as class. Examples: “application, system, data, computer, etc…”

C-Rule3: If a concept is related to Location name, People name, then ignore as a class. Examples: "John, Ali, London, etc…"

C-Rule4: If a concept is found in the high level of hyponyms tree, this indicates that the concept is general and can be replaced by a specific concept, then ignore as class.
Examples: "user, object, etc."

C-Rule5: If a concept is an attribute, then ignore as a class.
Examples: "name, address, number"

C-Rule6: If a concept does not satisfy any of the previous rules, then it's most likely a class.

C-Rule7: If a concept is noun phrase (Noun+Noun), if the second noun is an attribute then the first Noun is a class. The second noun is an attribute of that class. Examples:
"Customer Name" or "Book ISBN"

C-Rule8: if the ontology (if-used) contains information about the concept such as relationships, attributes, then that concept is a class.

2) Attribute Identification Rules:

A-Rule1: If a concept is noun phrase (Noun+Noun) including the underscore mark "_" between the two nouns, then the first noun is a class and the second is an attribute of that class. Examples "customer_name", "departure_date".

A-Rule2: If a concept can has one value, then it's an attribute. Examples:"name, date, ID, address". Based on A-Rule2, we collected and stored a predefined list including the most popular attributes to be used as a reference in RACE system.

3) Relationship Identification Rules:

R-Rule1: For each concept if {hypernyms_list} contains a concept (CT2) which have a hypernyms (HM) which is lexically equal to CT, then CT2 "is a kind of**"**CT. Then save result as Generalization relationship.

R-Rule2: If the concept is verb (VB), then by looking to its position in the document, if we can find a sentence having (CT1 - VB – CT2) where CT1 and CT2 are classes, then (VB) is an Association relationship.

R-Rule3: If the concept is verb (VB) and satisfies R-Rule2, and the concept is equal to one of the following {"consists of", "contain", "hold, "include", "divided to", "has part", "comprise", "carry", "involve", "imply", "embrace"}, then the relationship that discovered by that concept is Composition or Aggregation. Example: "Library Contains Books" then the relationship between "Library" and "Book" is Composition relationship.

R-Rule4: If the concept is verb (VB) and satisfies R-Rule2, and the concept is equal to one of the following {"require", "depends on", "rely on", "based on", "uses", "follows"} , then the relationship that discovered by that concept is the Dependency relationship. Example: "Actuator uses sensors and schedulers to open the door", then the relationships between ("Actuator" and "sensor"), ("Actuator" and "Scheduler") are the Dependencies relationships.

R-Rule5: Given a sentence in the form CT1 + R1 + CT2 + "AND"+ CT3 where CT1, CT2, CT3 is a classes, and R1 is a relationship. Then the system will indicate that the relation R1 is between the classes (CT1, CT2) and between the classes (CT1, CT3).

R-Rule6: Given a sentence in the form CT1 + R1 + CT2 + "AND NOT"+ CT3 where CT1, CT2, CT3 are classes, and R1 is a relationship. Then the system will indicate that the relation R1 is only between the classes (CT1, CT2) and not between the classes (CT1, CT3).

C. From Natural Language to Object Oriented Using the Natural Language Processing System LOLITA by L. Mich

Mich L. [5] proposes a NLP system, which generates an object model automatically from natural language. This approach considers nouns as objects and use links to find relationships amongst objects. LOLITA system is built on a large scale Semantic Network (SN) that does not distinguish between classes, attributes, and objects. This approach is limited to extract objects and cannot identify classes

D. A Taxonomic Class Modeling Methodology for Object-Oriented Analysis by Il-Yeol Song, Kurt Yano, Drexel University, USA

Song et al. [6] propose a TCM –taxonomic class modeling methodology for class identification. This method only identifies the classes from the requirements. Starting with problem statement, it incorporates the noun analysis, class categories, English sentence structure rules, checklists, and other heuristic rules for modeling This methodology help us to discover three types of classes: (1) classes represented by nouns in the requirement specifications, (2) classes whose concepts were represented by verb phrases, and (3) hidden classes that were not explicitly stated in the problem statement.

It works as follows. Candidate classes are listed by extracting nouns from the document. Now with the help of class elimination rules spurious classes are eliminated. English Sentence Structure Rules and other Heuristics are applied and a collection of domain classes are formed. Also hidden classes are discovered using pre-defined categories. By analyzing verb phrases and prepositional phrases transformed classes are identified. Finally, it is reviewed using domain knowledge. In this methods domain ontology

is used to refine the results. The union of all the identified classes gives the final set of classes in the class diagram.

III. CLASS DIAGRAM EXTRACTION APPROACH

In this section, we discuss our approach of class diagram extraction in detail. Section A describes the architectural design of our approach. Section B gives the detailed algorithm to implement our approach. Section C gives the general idea about how we are going to implement the system. Section D is a case study and in Section E a comparison is made with other approaches. Section F discusses merits and demerits of our approach.

In our approach, we use domain ontology as a source for identifying classes, attributes and relationships. Initially we parse the entire document with the help of parsers. Then with the help of POS taggers the entire document is tagged. Thus nouns, noun phrases and verbs are extracted from the document. Now these extracted nouns and noun phrases are the main source for class and attribute identification. Also verbs are the candidate for relationship and operation identification.

Noun verb noun triplets are generated from the functional requirements i.e. nouns linked to each verb is identified. This helps us to identify between which classes the relationship exists or to which class the attribute belongs.

We use domain ontology to determine whether each one belongs to class, attribute operation or inter class relationship. If search in the ontology fails then we further process the extracted nouns, noun phrases and verbs to identify classes, attributes, operations and inter class relationships.

The following table shows a mapping between web ontology language (a language used to formally represent ontology) components and UML class diagram components [8]. The major constructor of OWL are DataProperty, Object property, Class and Axioms. The corresponding constructor in UML class diagram is Attribute, Relationship, Class and Subclass respectively.

TABLE I. MAPPING BETWEEN OWL AND CLASS DIAGRAM

Owl components	UML class diagram components
DataProperty	Attribute
ObjectProperty	Relationships
Class	Class
Axioms	Subclass, Disjoint etc

If a noun is specified as DataProperty in the ontology, it links an individual to a data value. So it can be mapped to an attribute in our class diagram. If a noun is specified as ObjectProperty in the ontology, it links an individual to another individual. So it can be mapped to the relationship between two classes in our class diagram.

A. *Architectural design*

In this section the architecture model of proposed methodology is described. The major blocks are SRS Parser, NVN Triple extractor, OWL Parser, Classes and Attributes extractor, Relationships and Operations extractor and Integrator. Each module is described below.

SRS Parser and tagger: It parses and tags the functional requirement specification statements and identifies all the verbs and nouns from it. Now nouns serves as a source of attribute and class identification. Also verbs serve as a source of operations and relationship identification.

NVN Triple extractor: It generates noun verb noun triplets from the functional requirements. Thus nouns attached to each verb are identified. Nouns can be converted to classes or attributes and verb can be converted to relationships or operations. So from the NVN triplets, if both the nouns attached to verb are class then the verb represents a relationship between those classes. Also if one is a class and other an attribute then the verb represents an operation of that class.

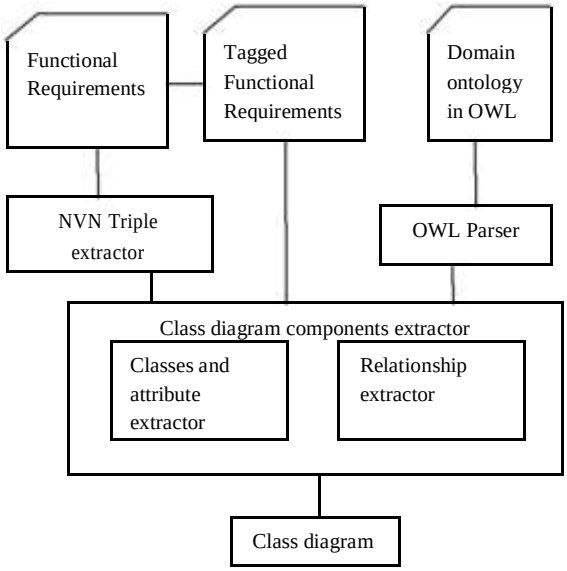

Fig. 1 An architectural design for proposed approach

OWL Parser: It parses the OWL ontology. In ontology domain specific conceptual terms are specified as classes, inter term relationships are specified as ObjectProperty and attributes are represented as DataProperty. So with the help of OWL Parser these are extracted and stored.

Classes and Attributes extractor: It extracts the components like classes and attributes from the nouns and noun-phrases. Domain ontology is used in this step. Attributes are specified as DataProperty in OWL. If domain ontology does not contain any information about the nouns and noun phrases under consideration, then the tagged nouns and noun phrases are again processed. It is described in detail in the following section.

Relationships and Operations extractor: It identifies the UML class diagram components like operations and inter class relationships from the tagged verbs. Here also domain ontology is used. Relationships are specified as ObjectProperty in OWL. If the ontology does not contain any information about the verbs under consideration then requirements are again processed to understand relationships. It is described in detail in the following section.

Class diagram: All the components identified from the previous stages, i.e., classes, attributes, relationships and operations are integrated and final class diagram is generated.

B. *Algorithm for class diagram extraction*

Now we discuss the detailed actions to be carried out in order to extract the class diagram from natural language requirements. It includes mainly four modules. They are initial parsing, tagging and NVN triplet extraction of sentences, parsing of ontology represented in OWL, identification of classes and attributes and finally identification of relationships and operations.

First module outputs the tagged SRS document from which nouns, noun phrases, verbs and NVN triplets can be identified (Fig.2). Next module is OWL parser which parses the ontology represented in OWL and thus sub class relationships, ObjectProperties and DataProperties are extracted (Fig.3). Now the nouns and noun phrases becomes the candidate for classes and attributes identification (Fig.4). Noun-Verb-Noun triplet becomes candidates for relationship and operation identification (Fig.5). In the last two steps domain ontology is used to explore the important concepts of the domain.

1) *Algorithm to identify noun, noun phrase and verbs in the functional requirement specification:*

Pseudo code:
Input: Functional Requirements
Output: Tagged sentences and noun-verb-noun triplets

1. WHILE(end of functional requirement specification reaches)
   i) Parse each sentence
   ii) Tag parsed sentence with the help of POS taggers.
   iii) From the tagged sentences extract nouns, noun phrases and verbs.
   iv) Store nouns to a file called noun.
   v) Generate noun-verb-noun triplets for each sentence in the functional requirements in order to identify to which nouns each verb is linked.
   vi) Store triplets to file triplets.
2. ENDWHILE

Fig. 2 Algorithm to identify noun, noun phrase and verbs

Input to this phase is our natural language functional requirements. It is parsed with the help of parsers. Now with the help of parts of speech taggers each sentence is tagged. From the tagged sentence we can identify nouns, noun phrases and verbs. Also noun-verb-noun triplets are formed with help of semantic role labelers. Nouns and noun phrases are input to second stage which is classes and attribute extractor. Noun- Verbs-Noun triplets are input to third stage which is relationships and operations extractor. These nouns are stored in a file named as nouns and NVN triplets in a file called triplets.

2) Algorithm to parse the OWL ontology:

Pseudo code:
Input: OWL ontology
Output: Classes, Subclasses, ObjectProperties and DataProperties.

1) WHILE(entire OWL ontology is processed)
   a. Generate a class hierarchy from the ontology which helps to identify class-subclass relationships.
   b. Extract ObjectProperty from the OWL which helps to identify association relationships
   c. Extract DataProperty from the OWL which helps to identify attributes.
2) ENDWHILE

Fig. 2 Algorithm to parse OWL ontology.

3) Algorithm to identify Classes and attributes:

Pseudo code:
Input: Nouns and noun-phrases from tagged sentences
Output: Classes and Attributes

1) WHILE (all the nouns and noun phrases processed)
   a) IF noun is present in the parsed ontology file
      i) IF it is presented as a class in the file
         i) Mark noun as a class.
         ii) Extract all the associated information like its attributes (represented as DataProperty), association relationship (represented as ObjectProperty) with other classes and its immediate sub classes.
         iii) Store all the extracted information along with its type in a temporary memory.
      ii) ENDIF
      iii) IF it is stored as an attribute in the temp memory
         i) Mark noun as an attribute of related class
      iv) ENDIF
   b) ELSE
      i) IF the noun candidate pass all the nine rules given below
         i) It is a class
      ii) ELSE
         i) It can be an attribute or operation as given by the rules.
      iii) ENDIF
      iv) IF noun has only one property to remember
         i) It is a attribute of the related class
      v) ELSE
         i) It is a class.
      vi) ENDIF
      vii) IF identification of the noun relies on another noun
         i) First noun is an attribute of second which is a class
      viii) ENDIF
2) ENDWHILE

Fig. 4 Algorithm to identify Classes and attributes

Process the nouns and noun phrases to extract classes and attributes. Initially domain ontology is searched. Since the major concepts of the domain are described in the ontology, it helps to determine whether the noun represents a class or an attribute. If it is defined as a class in the ontology, all the information like its attributes and its relationship with other classes can be extracted. Thus ontology helps to extract properties which are not explicitly stated in the functional requirements. If ontology does not contain any terms that matches with the noun under consideration, it is a new term that need to be included in the system to be developed. So process them separately.

To process the nouns not present in the ontology, we use the Class Elimination Rules proposed by Song *et al*. [6] in TCM –taxonomic class modeling methodology. If the noun is anyone of the following category, it can be eliminated as a class. The rules are given below.

TABLE II. RULES TO ELIMINATE NOUN AS A CLASS

Category		Description	Actions
Classes	Redundant classes	If two nouns represent the same abstraction	keep the most descriptive one
Classes	Irrelevant classes	The nouns have nothing to do with the problem to be solved	The noun is beyond the scope of the problem being modeled. So ignore the noun as class
Classes	Vague classes	The nouns have ill-defined or too broad scope	Ignore the noun as class
Implementation constructs		The nouns represent an implementation-related class such as set, string, or algorithm.	These implementation classes can be added at the design or implementation stages, but not at the conceptual level.
Meta-language		The noun is used to describe and explain requirements and the system at a very high level. Examples are systems, information etc	Ignore the noun as class
Operations		The nouns represent operations. For example, fine calculation is a noun form of an operation called calculates fine.	Add noun as an operation of the associated class
Attributes		The nouns represent a text or a number. For example, name, age, and phone number represent attributes that carry a value.	If the noun has one property to remember then it is an attribute else it is a class or if the identification of the noun relies on another noun the dependent noun is an attribute of the other noun.
Values		The nouns represent a value itself.	Ignore noun as a class

3) Algorithm to identify relationships and operations:

Pseudo code:

Input: Verbs from tagged sentences and NVN triplets
Output: Operations and relationships

1) WHILE all verbs are processed
   a)IF verb is present between two classes
      i) IF one class is represented as subclass of other class in the temporary memory location used in the previous stage
         (a) Verb represents a Generalization relationship
      ii) ENDIF
      iii) IF one class is represented as an association of other class in the temporary memory location used in the previous stage
         a) Verb represents an association relationship
      iv) ENDIF
      v) IF verb is a word with meaning “consist of”
         (a) Verb represents a Composition or Aggregation relationship
      vi) ENDIF

vii) IF verb is a word with meaning "uses"
(a) Verb represents a Dependency relationship
viii) ENDIF
b) ELSE
i) IF verb is linked with a single class
(a) Verb represents an operation of the linked class
ii) ENDIF
c) ENDIF
2) ENDWHILE

Fig. 5 Algorithm to identify relationships and operations

Process the verbs of the natural language statements, in order to extract operations and relationships. If verb is present between two identified classes then it must be relationship between the two classes. Otherwise it is an operation of the linked class. In the previous stage the association and generalization relationship between classes is stored in a temporary memory. So extract that information from the temporary memory. If verb is a word with meaning "consist of" it represents a Composition or Aggregation relationship. If verb is a word with meaning "uses" it represents a Dependency relationship

C. *Implementation*

To implement the above algorithm, the system will have to use many external programs and resources. Here a POS tagger and parsers will be used as external resources. Java native interfaces help us to interface all these recourses with our main java application. Domain ontology and SRS document are the input to our system. Fig. 6 illustrates the components of the implementation

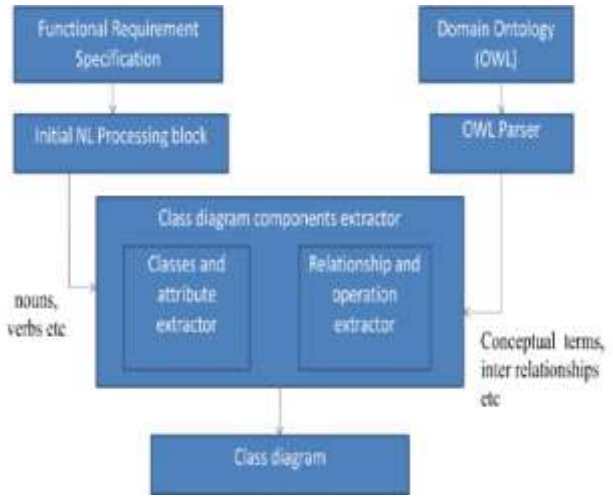

Fig. 6 Implementation of the proposed approach

Functional Requirement Specification: This is the main input to our application. This document contains end users requirements specified using natural language statements.

Domain ontology: This is another input to the application. It formally represents knowledge as a set of concepts within a domain and may be used to describe the domain [2]. All the core concepts and their relationships are represented in the ontology in a formal way which helps to extract the classes, attributes and relationships between classes from our functional requirement specification. With the help of OWLAPI [10] we parse the OWL ontology. The OWL API is a Java API and reference implementation for creating, manipulating and serializing OWL ontology. The OWL API is open source and is available under either the LGPL or Apache Licenses.

Parser and tagger: Parsing [7] is the process of analyzing a text, made of a sequence of tokens, to determine its grammatical structure with respect to a given formal grammar. Part-of-speech tagging (POS tagging or POST), also called grammatical tagging or word-category disambiguation, is the process of marking up a word in a text (corpus) as corresponding to a particular part of speech, based on both its definition, as well as its context i.e. relationship with adjacent and related words in a phrase, sentence, or paragraph. A simplified form is the identification of words as nouns, verbs, adjectives, adverbs, etc. To perform parsing and tagging, number of java based natural language processing tools are already available. One of them is OpenNLP. The Apache OpenNLP library is a machine learning based toolkit for the processing of natural language text. It supports the most common NLP tasks, such as tokenization, sentence segmentation, part-of-speech tagging, named entity extraction, chunking, parsing, and co reference resolution.

Noun-verb-noun triplet extractor: We need to identify the nouns attached to each verb in order to map them correctly to relationships or operations. NVN extraction is done with the help of semantic role labeler. Semantic role labeling(SRL), sometimes also called shallow semantic parsing, is a task in natural language processing consisting of the detection of the semantic arguments associated with the predicate or verb of a sentence and their classification into their specific roles. For SRL a tool called SENNA [11] is used. SENNA is software distributed under a non-commercial license, which outputs a host of Natural Language Processing (NLP) predictions: part-of-speech (POS) tags, chunking (CHK), name entity recognition (NER), semantic role labeling (SRL) and syntactic parsing (PSG). SENNA is written in ANSI C, with about 3500 lines of code

Class diagram components identification: This is the main module which implements the entire algorithm specified in Fig. 2, Fig. 3, Fig. 4 and Fig. 5. Inputs to the module are Functional Requirement Specification and domain ontology. The module outputs the major components of the class diagram.

Class diagram: This is the final class diagram generated. It includes classes, attributes and relationship between classes like generalization, association and composition.

C. *Case study*

Here we utilize an open source ontology ka.owl which defines the major concepts that comes under area academic research center.

1) Academic research organization management requirements

"Employees are head of some projects and students works at some projects. Employees are uniquely identified by emp_ID and students by stud_ID. Projects are financed by university. University consists of a number of departments. Three types of employees are academic staff, technical staff and office staff. All academic related activities are handled by academic staff. Technical staff provides technical support and office staff carries all non academic activities"

2) A sample class hierarchy from ka.owl ontology

Fig 7 shows the sample hierarchy of knowledge acquisition ontology.

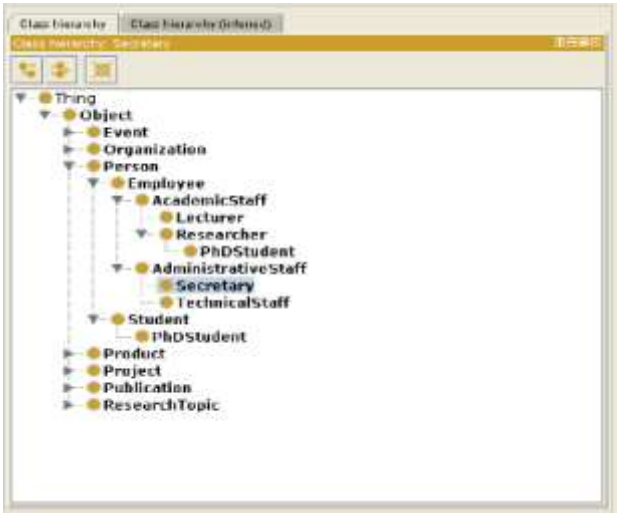

Fig. 7 Class hierarchy of knowledge acquisition ontology

Knowledge acquisition ontology is free open source ontology defines concepts from academic research and contributed by Ian Horrocks . In the ontology event, organization, person, product, project, publication, and research topic are represented as major concepts. These may or may not be relevant with respect to our requirements. The relevant classes are identified by comparing our requirements in the functional requirement specification with domain ontology. Now the associated information of the classes is also retrieved from the ontology. Thus ontology helps to extract deepest information about major concepts.

The above snapshot is taken from the tool Protégé. Protégé is a free, open-source platform that provides a growing user community with a suite of tools to construct domain models and knowledge-based applications with ontology. At its core, Protégé implements a rich set of knowledge-modeling structures and actions that support the creation, visualization, and manipulation of ontology in various representation formats. Protégé can be customized to provide domain-friendly support for creating knowledge models and entering data.

With the help of OWLAPI we can parse the knowledge acquisition ontology. Thus the major conceptual terms, inter term relationships and attributes of each terms described in the ontology can be extracted. These are used in the later stage of class diagram components identification i.e. to determine whether a term in the requirement belongs to a class, attribute, relationship or operation.

3) Class diagram generated

From the ontology, we get the information that employee and students are subclass of person. Also academic staff and administrative staff are the subclass of employee. Properties associated with each concept can also be extracted from the ontology.

From the requirements employee is tagged as noun which is defined as a core concept in our ontology. So the entire information associated with the concept "employee" can be extracted from the ontology. Now from the requirements student is tagged as a noun. From the ontology it is clear that both student and employee share a lot of common attributes. So with respect to the ontology they are represented a subclass of the class person. In the requirements it is specified that emp_ID and stud_ID are used to uniquely identify employee and student respectively. It is added to the student class and employee class as its attributes.

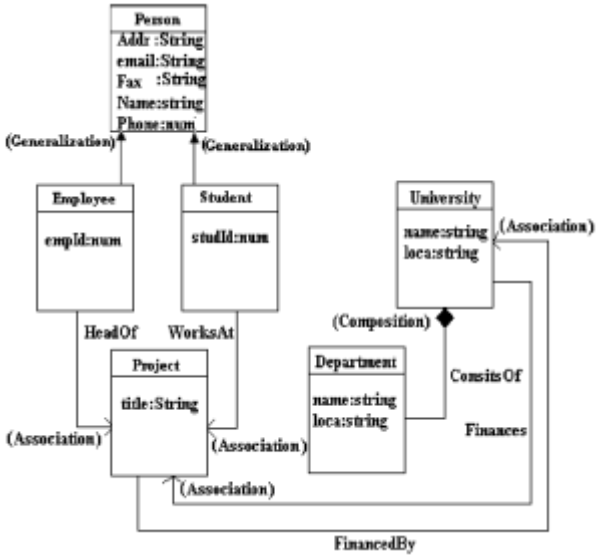

Fig. 8 The class diagram generated

The association relationship "works at" between student and project can be extracted from ontology. Similarly "head of" relationship between employee and project can also be extracted from the ontology. Since university consists of number of departments, a composition relationship exists between university and department. The following figure shows the sample class diagram generated for the above requirements.

D. *Comparison with other approaches*

In this section a comparison is made between our methods with two other methodologies.

A methodology proposed by Zhou and Zhou [3] that uses both NLP and domain ontology, process the natural language requirements with the help of some linguistic patterns. Also domain ontology is provided as input in some specific format and its scope is too limited. In our approach we are providing the ontology in the standard ontology representation language, OWL. The Web Ontology Language (OWL) is a family of knowledge representation languages for authoring ontology with some formal semantics.

Mohd Ibrahim, Rodina Ahmad [4] proposes a methodology that also uses NLP and domain ontology. In this method, based on the linguistic structure of the sentences, the natural language statements are processed and finally domain ontology is used to refine the result. Anyhow In our approach, domain ontology is used initially to fully explore the identified classes. Domain ontology gives an idea about the identified class's major attributes, operations and relationships with other classes. If ontology does not contain any description of the term that needs to be finding out, then only we go for other natural language level processing to understand the concept completely.

E. *Merits and demerits*

The major advantage of our approach is that, we use the existing ontology which is represented in OWL. As per W3C standards, OWL is a standard language for representing ontology. W3C standards [9] define an Open Web Platform for application development that has the unprecedented potential to enable developers to build rich interactive experiences, powered by vast data stores that are available on any device. Also domain ontology is used in the initial stage of processing the requirements to understand the major concept of the particular domain. One drawback of our approach is that the domain ontology, which is the core of this method, needs to be available for the particular domain for which the system is to be developed. But once they are developed it can be reused for other applications also.

IV CONCLUSION

Domain ontology contains all the core concepts for a particular domain. So it can be used as a source to identify major classes, relationships and attributes. In our method we make use of the benefits supported by domain ontology. Our ontology is represented in the OWL which is a standard ontology representation language. If ontology contains description about the terms in the requirement specification, then we can retrieve all its attributes and relationship with other concepts in that domain. If ontology does not contain the terms, then our natural language statements are further processed to identify classes, attributes and relationships.

The major advantage of this method is that, here the existing ontology is used which are represented in OWL. As per W3C standards, OWL is a standard language for representing ontology. W3C standards define an Open Web Platform for application development that has the unprecedented potential to enable developers to build rich interactive experiences, powered by vast data stores that are available on any device. Also domain ontology is used in the initial stage of processing the requirements to understand the major concept of the particular domain.

Nowadays number of ontology is available online. So we can reuse the existing ontology if they are already developed for the required domain. Otherwise we can easily develop our own OWL ontology with the help of the tool Protégé. So once developed they can be reused for other application also.

Major steps of this method are

- Identification of nouns and verbs from requirement specification statements
- Generation of c with the help of semantic role labelers. We used a tool called SENNA for semantic role labeling
- Parsing of the domain ontology. It was done with OWL API.
- Generate classes and attributes from the list of noun and noun phrases by linking them with parsed ontological terms.
- Generate relationships and operations from relationships and operations.

So a method for class Model generation from functional requirement specification is presented. Here we used domain ontology represented in OWL format to identify classes, their attribute and inter class relationships. A detailed algorithm to implement our approach is also presented.

VIII REFERENCES

[1]. Booch. G., Object-Oriented Analysis and Design with Applications, *2nd Ed.*, *Benjamin Cummings*, 1994.

[2]. Gruber, Thomas R., A translation approach to portable ontology specifications", *Knowledge Acquisition* 5 (2): 199–220, (June 1993).

[3]. Xiaohua Zhou and Nan Zhou, Auto-generation of Class Diagram from Free-text Functional Specifications and Domain Ontology, *Artificial Intelligence*, 2004.

[4]. Mohd Ibrahim, Rodina Ahmad, Class diagram extraction from textual requirements using Natural language processing (NLP) techniques, *second international conference on computer research and development*, 2012.

[5]. L. Mich, NL-OOPs: From Natural Language to Object Oriented Using the Natural Language Processing System LOLITA., *Natural Language Engineering*, 2(2, pp.161-187), 1996.

[6]. Song, Il-Yeol, et al. A Taxonomic Class Modeling Methodology for Object-Oriented Analysis , *In Information Modeling Methods and Methodologies: Advanced Topics in Databases Series, Ed*, pp. 216-240, 2004.

[7]. Daniel Jurasky, Daniel H. Martin Speech and language processing: An introduction to natural language processing, computational linguistics, and speech recognition.

[8]. Soraya Setti Ahmed and Sidi Mohamed Benslimane, Reverse Engineering Process for Extracting Views from Domain Ontology.

[9]. Deborah L. McGuinness and Frank van Harmelen, OWL Web Ontology Language Overview, Editors, *W3C Recommendation*, 2004

[10]. Matthew Horridge, Sean Bechhofer. The OWL API: A Java API for OWL Ontologies. *Semantic Web Journal 2(1), Special Issue on Semantic Web Tools and Systems*, pp. 11-21, 2011.

[11]. R. Collobert, J. Weston, L. Bottou, M. Karlen, K. Kavukcuoglu and P. Kuksa. Natural Language Processing (Almost) from Scratch, *Journal of Machine Learning Research(JMLR)*, 2011