UDRA: Reflecting Natural Language Text in to UML Diagrams

AmrutaAmune[a] and Radhakrishna Naik[b]

[a]*Department of Computer Engineering, MIT, Aurangabad*
[b]*Department of Computer Engineering, MIT, Aurangabad*

ABSTRACT

Today the software researchers and the industry specialists are paying more attention towards the automating the job of requirement analysis. As we all know requirement analysis is the preliminary step in software development process. For the same requirement analyst spends significant amounts of time to get the requirements. Though there exist various methodologies and tools to facilitate the automatic reflection of user requirement in to UML diagrams, but none of them can be used practically due to highly complex in nature. We present a new methodology for generating UML class diagrams or models from natural language problem statement or requirement specification. We have named our methodology as UML Diagram Extraction through Requirements Analysis, which uses natural language and domain ontology techniques for the extraction of various UML diagrams. It is a desktop tool to support requirements analysts and SE students in analyzing textual requirements, finding core concepts and its relationships, and step by step extraction of the UML diagram. Proposed method not only enhances the productivity but also accelerate the time to develop the software.

Keywords:
Natural language processing (NLP),
Domain Ontology,
UML Class Diagram,
Object oriented design (OOD.

1. Introduction

Requirement analysis is the preliminary step in software development process. The requirements stated by the clients are analyzed and an abstraction of it is created which is termed as requirements model. If any mistake or misconception occurs during analyzing requirements then it may leads to some blunder mistakes in software & thereby reducing efficiency of some intended functionality of software. Therefore requirement analysis is considered as most important step in the development of software. Unified Modeling Language (UML) [1] models are helpful for understanding the problems, communicating with application experts and preparing documentation. The static design view of the system can be modeled using a UML class diagram. System requirements stated by the user are usually in natural language form despite a wide variety of formal languages and UML.

If any mistake or misconception occurs during analyzing requirements then it may leads to some blunder mistakes in software & thereby reducing efficiency of some intended functionality of software. Therefore requirement analysis is considered as most important step in the development of software. So instead of having manual tedious process of requirement analysis, we require some automated tool that will help software analyst in analyzing this requirement and giving clear idea about core concepts in the development of software.

To analyze a given text, the most Natural Language Processing (NLP) [2] systems are based on the following levels: Morphological level, lexical level, syntactic level, semantic level, discourse level and pragmatic level [3]. Domain ontology has also been widely used to improve the efficiency of concepts identification. It models a specific domain, which stands for a part of the world. Using this kind of ontologies, organizations, enterprises or communities describe the concepts in their domain, the relations between those concepts, and obviously the instances that are the actual things that fill its structure.

The main goal of the proposed work is to investigate how Natural Language Processing techniques and Domain Ontology can be exploited to support the Object-Oriented Analysis process. This study must accept, as an input, textual data expressed in natural language and will efficiently able to extract the basic elements necessary for creating a UML diagram from clearly specified user requirements.

The rest of this paper is organized as; section 2 highlights the related works along with their downsides, section 3 discusses our proposed model to overcome the downsides of the existing work, section 4 discuss with performance analysis. Section 5 shows the result with case study and section 6 gives the conclusion followed by references

2. Related work

In the last two decades have seen growing interest in AI based Software Engineering technology. Our review of this area shows that there have been only few approaches that provide automated tools for supporting the early stages (mainly analysis and design) of software development.

Few NLP (natural language processing) based tools are there to analyze the requirements statement and to support object oriented modeling. [4] Proposed a method for using natural language text for creating object oriented models. It produced static analysis and design modules which required high user intervention for making decisions. According to him nouns can be taken as classes and verbs constitute the functions or methods for classes. [5] Suggested that nouns can be considered as both classes and attributes. After that [6] proposed that association relationship between different classes can be pointed out by verbs. [24] Also presented a semi-natural language to automatically generate object models from natural language text. A prototype tool [7] is used by it which created static and dynamic models from the problem statement. Problems related to natural language were also solved by the work done by [10]. [8] also has done requirement analysis by matching the input text with predefined statements The above methods are not automatic and the analyst has to be involved to take decisions to create models. Some other studies [9] [10] also have been conducted to derive class diagrams from natural-language documents [2] by

using natural language processing (NLP) techniques. The success of all the above works depends on the quality of the document. The description of the document should contain all the important details of the problem; otherwise the correct class models are not extracted.

The information obtained from the above prescribed works served as a framework for suggesting several Computer Aided Software Engineering (CASE) tools [10] for model creation from English requirements. REBUILDER UML [12] is a CASE tool presented by Oliveira and used natural language components for modeling purpose. This approach was based on Case- Based Reasoning and it deals only with class diagrams and needs continuous up gradation of case-base. Another tool to generate a class model was LOLITA [13]. It identifies only objects and the classes, and attributes cannot be distinguished. REVERE [14] makes use of a lexicon to clarify the word senses. The main disadvantage of these CASE tools is that the object oriented elements were not automatically extracted from requirement specification. The user has to be involved to identify classes and their constituents.

LIDA [15] helps analysts develop object-oriented models of a domain, using a subset of UML. It consists of two main interface components: Text Analyzing Environment and a Model Editing Environment. The problem with LIDA is, it needs more user interaction while generating diagrams, as it identifies just the list of nouns, verbs and adjectives and the developer has to decide which word goes in classes or attributes or operations. So LIDA depends on knowledge of problem domain.

CM-Builder [16] is another tool that uses robust Natural Language Processing techniques to analyze software requirements texts written in English and builds an integrated discourse model of the processed text, represented in a Semantic Network. This Semantic Network is then used to automatically construct an initial UML class. It is a modular NL-based CASE tool which performs domain independent OO analysis. Class diagram generated by CM-Builder doesn't include operations.

UCDA [17] tool overcomes this problem of CM-Builder. It uses a freely available natural language parser and Rational Rose's extensibility interface to support the automation of the Object Model Creation Process (OMCP). It identifies the analysis model of the class and can just visualize the UML diagrams in Rational Rose, which is not convenient for those users who have not installed Rational Rose on their systems.

CIRCE [18] can also be considered as good attempt in extracting UML models. . It is an environment for the analysis of natural language requirements. It helps in the elicitation, selection and validation of software requirements. Here CICO is main tool that is considered as front end for other components which recognizes the NL sentences & extracts some facts from them. These facts are then handed to the remaining tools for graphical representation and analysis. But it is not so efficient as it lacks in its language handling capability.

MOVA [19]tool allows the user to draw UML class and object diagrams, write and check OCL invariants, write and evaluate OCL queries also allows us to write and evaluate OCL metrics. MOVA Meta model is only a subset of the UML Meta model; it does not support the full OCL syntax. Finally, class and object diagrams are saved in a MOVA specific XML format, which precludes the models for being exchange with other tools.

SUGAR [20] is aimed at assisting the developer in generating static UML diagram. The main lacuna in SUGAR is that, it poses constraint on specifying requirements like each sentence should be expressed in active voice. Accuracy of Candidate Class identification, relationship is limited & It needs human interaction (such as at the time of eliminating irrelevant classes) while deciding the candidate classes which may sometimes mislead if there is lack of domain knowledge.

Relative Extraction Methodology [21] tries to resolves SUGAR problems. Even though it seems efficient, it lacks in relationships specification such as does not include advanced relationships like aggregation and dependency between classes. The multiplicities between the classes were also not considered there.

UMGAR [22] tool can be considered as more advanced version of SUGAR & Relative extraction methodology. It generates UML models like the use case diagram, analysis class model, collaboration diagram & design class model from natural language requirement using efficient NLP tools. The requirement of XMI import facility makes its usage limited.

Object Oriented Software Modeling Using NLP Based Knowledge Extraction [23] is a recent technique that presents a natural language processing based automated system for NL text to OO modeling the user requirements and generating code in multi-languages. First the NL text is semantically analyzed to extract classes, objects and their respective, attributes, methods and associations. Then UML diagrams are generated on the bases of previously extracted information. The designed system also provides with the respective code automatically of the already generated diagrams. The designed system provides a quick and reliable way to generate UML diagrams to save the time and budget of both the user and system analyst. Accuracy of generating diagrams of this system was about 80% to 85%. It can be enhanced up to 95% by improving the algorithms and inducing the ability of learning in the system

A clear analysis of previous works infers that no good attempt has been made for the construction of UML diagrams from the NL text and all the methods are highly complex. No tools can be used practically due to lacunas it contain are listed below.

1. There is no framework for auto-generation of complete class diagrams from free-text functional specification documents.
2. Most of earlier tool doesn't allow user to visualize UML diagrams without installing any UML representation tool like Rational Rose.
3. Some existing tools require human interactions for automatic development of a Class model along with associated attributes & operations.
4. Advanced relationships like Generalization, Association, Composition, aggregation and dependency does not provided in existing tools.

The significance of our method is to resolve each of these above cited problems. UDRA system takes informal requirement document or SRS as an input & generates UML diagrams like Class diagrams, Object diagrams, Package diagrams & Deployment diagrams. It generates complete class diagram including attribute specification, operations specifications as well as all advanced relationships like generalization, association, compositions etc. While using UDRA tool it is not mandatory for users to install diagram visualizing tool, it can be easily handled by UDRA built in system & it's GUI. The GUI provided by UDRA is user friendly that allows user to modify, print or edit generated diagrams.

3. Proposed Approach For UDRA

In the previous section, we have reviewed the most recent works. UNDRA is a desktop instrument to assist requirement analysts and SE students in analyzing textual requirements, finding core concepts and it's relationships, & step by step extraction of the UML diagrams like class diagram ,Object diagram, Deployment Diagram & Package Diagram .It is based on natural language techniques & domain ontology.

This approach synthesizes several different class modeling techniques under one framework which integrates the noun analysis method, class categories, English sentence structures, check lists, and other heuristic rules for modeling. The aim of our approach is to efficiently apply NLP and domain ontology techniques to achieve a fast and accurate analysis result.

Proposed System is divided into three modules as follows. Module 1 is the Concept Extraction engine. Were as 2 Module is Class identification which involves identification of candidate classes, packages, nodes & associated relationships & attributes by using Heuristic rules, and 3 Module is Concept Management module which handles graphical user interface of UDRA tool.

The input files include free-text functional specifications and structured domain ontology while the output is UML diagram components including classes, object, Nodes and package attributes of each, and inter-class relationships. The inter-class relationships can be classified into generalization, aggregation, and dependency. Fig 1 shows architectural view of UDRA

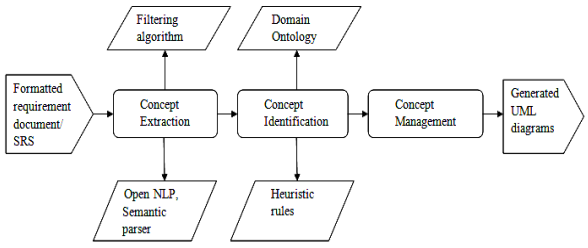

Figure 1: Architecture of UDRA system

3.1. Concept Extraction

In this we are trying to retrieve candidate concepts that can be considered as classes, packages, nodes, objects etc & while achieving this ambiguous & extraneous information is discarded.

3.1.1. Open NLP Parser

For natural language processing us can use different tools like Sentence Detector, Tokenizer, Part-Of-Speech tagger (POS), Tree bank parser etc. Here we have used Open NLP Parser [24]. It is an open-source and re-usable algorithm. It provides our system with lexical and syntactic parsers. Open NLP POS tagger (lexical) takes the English text as input and outputs the corresponding POS tags for each word; On the other hand, OpenNLPChunker (syntactic) chunks the sentence into phrases (Noun phrase, verb phrase, etc.) according to English language grammar.

3.1.2. UDRA stemming algorithm

Stemming/filtering is a technique that abbreviates word by removing affixes and suffixes [24]. In UDRA system, it is very important to return words back to its base form; this will reduce the redundancy and increase the efficiency of the system. To perform the stemming, we implemented a new stemming algorithm. Based on the stemming result, we find that our stemming algorithm is efficient and sufficient to be used in the morphological analysis of requirements in UDRA system. Our stemming algorithm is simple and re-usable.

The algorithm is presented below.

Step1 : Use the requirements document as input.

Step2 : Identify the stop words and save the result as {Stopwords_Found} list.

Step3 : Calculate the total number of words in the documents without the stop words, the number of occurrences of each word, and then calculate the frequency of each word, as in.

Step4 : Use UDRA stemming algorithm module to find the stemming for each word and save the result in a list.

Step5 : Use OpenNLP parser in [8] to parse the whole document (including the stop Words).

Step6 : Use the parser output to extract Proper Nouns (NN), Noun phases (NP), verbs (VB). And save it in {Concepts-list} list.

Step7 : Use Step2 and Step 6 to extract: {Noun phrases (NP) – {Stopwords_Found} And save results to {Concepts-list}.

Step8 : For each concept (CT) in {Concepts-list} if {synonyms_list} contains a concept (CT2) which have a synonym (SM) which lexically equal to CT, then CT and CT2 concepts are semantically related to each other.

Step9 : For each concept (CT) in {Concepts-list} if {hypernyms_list} contains a concept (CT2) which have a hyponyms (HM) which lexically equal to CT, then CT2 "is a kind of CT. Then save result as {Generalization-list}.

3.1.3. Word Net

Word Net [24] is used to validate the semantic correctness of the sentences generated at the syntactic analysis. It also enables users to display all hyponyms for a selected noun. We used this feature to verify Generalization relationship where a noun phrase is supposed to be 'a kind of' another noun phrase. Word Net can used to find semantically similar terms, and for the acquisition of synonyms.

3.2. Class Extraction Engine

This module uses different heuristic rules to extract the class diagram; However, We use domain ontology in this module to refine the extracted class diagram [24]. We can summarize the heuristic rules used as the following

3.2.1. Class Identification Rules

C-Rule 1 : If a concept is occurred only one time in the document and its frequency is less than 2 %, then ignore as class.

C-Rule 2 : If a concept is related to the design elements then ignore as class.

C-Rule 3 : If a concept is related to Location name, People name, then ignore as a class.

C-Rule 4 : 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.

C-Rule 5 : If a concept is an attribute, then ignore as a class.

C-Rule 6 : If a concept does not satisfy any of the previous rules, then it's most likely a class.

C-Rule 7 : 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.

C-Rule 8 : If the ontology (if-used) contains information about the concept such as relationships, attributes, then that concept is a class.

3.2.2. Attribute Identification Rules

We use the following rules for attributes identification.

A-Rule 1 : 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.

A-Rule 1 : If a concept can has one value, then it's an attribute.

3.2.3. Relationship Identification Rules

Using verb analysis as input, we can apply the following rules

R-Rule 1 : Using step 9 in the concept extraction engine (section 4.4), all the elements in the {generalization-list} will be transferred as Generalization (IS-A) relationship.

R-Rule 2 : 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-Rule 3 : 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-Rule 4 : 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.

R-Rule 5 : 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.

R-Rule 6 : 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-Rule 7 : 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).

3.3. Package Extraction Engine

At the first step, packages that extracted using the 'Package Extraction Engine'

P-Rule 1 : If a package is related to the functionality of elements then consider as package

P-Rule 2 : If a concept is occurred only one time in the document and its frequency is less than 2 %, then ignore as Package name.

P-Rule 3 : If the ontology (if-used) contains information about the concept such as relationships, then that concept is a Package.

3.3.1. Relationship Identification Rules

Using verb analysis as input, we can apply the following rules: Two Relationship in Package diagram as follows.

1. Import: A package import is defined as a directed relationship that identifies a package whose members are to be imported by another package
2. Merge: A package merge is a directed relationship between two packages that indicates that the contents of the two packages are to be combined.

R-Rule 1 : If the concept is verb (VB) i.e. Package, then by looking to its position in the document, if we can find a sentence having (VB1 –Importing – VB2) where VB1 and VB2 are package, then it is an Importing relationship.

R-Rule 2 : If the concept is verb (VB) i.e. Package, then by looking to its position in the document, if we can find a sentence having (VB1 –Merge – VB2) where VB1 and VB2 are package and contents of the two packages can be combined, then it is a merge relationship.

R-Rule 3 : Given a sentence in the form Pack1 + R1 + Pack2 + "AND"+ Pack3 where Pack1, Pack2, Pack3 are Packages, and R1 is a relationship. Then the system will indicate that the relation R1 is between the Package (Pack1, Pack2) and between the Package (Pack1, CT3).

R-Rule 4 : Given a sentence in the form Pack1 + R1 + Pack2 + "AND NOT"+ Pack3 where Pack1, Pack2, Pack3 are Packages, and R1 is a relationship. Then the system will indicate that the relation R1 is only between the Package (Pack1, Pack2) and not between the Packages (Pack1, Pack3).

3.4. Object Extraction Engine

This module uses the output of "Object extraction engine" module and applies different heuristic rules to extract the Object

diagram; However, We use domain ontology in this module to refine the extracted Object diagram. We can summarize the heuristic rules used as the following

3.4.1. Object Identification Rules

O-Rule 1 : If a concept is noun phrase (Noun+Noun), if the second noun is an Object/Instance name then the first Noun is a class name. The second noun is an Object/Instance name of that class.

O-Rule 2 : If a concept is occurred only one time in the document and its frequency is less than 2 %, then ignore as Object name as well as class name.

O-Rule 3 : If the ontology (if-used) contains information about the concept such as relationships, then that concept is a Package and relationship is Association or Direct relation between them.

4. Performance Analysis

This section presents the results of a practical performance evaluation of UDRA system. Although these techniques do not produce an optimal solution but the solution produced are sufficiently close to the optimal one and diagrams generated by it are simple, well established and understood.

4.1. Performance evaluation based on Recall, Precision & Over specification

In order to evaluate UDRA system performance, we have compared the models produced by the UDRA system with the models produced by human analysts. We have used three metrics for it that is used in CM-Builder's [16] performance evaluation. These are;

1. Recall: It focuses Completeness parameter; how much completed model system has produced. It can be given as,

   Recall $= N_{correct} / N_{key}$

   Where

   $N_{correct}$ - number of correct responses made by the system.

   N_{key} - number of information elements in answer key.

2. Precision: It focuses on Accuracy parameter; how much of the information produced by the system is correct. It can be given as,

   Precision $= N_{correct} / (N_{correct} + N_{incorrect})$

   Where $N_{incorrect}$ - refers to incorrect responses made by the system.

3. Overspecification: It measures how much extra correct information in the system response is not found in the answer key.

   Overspecification $= N_{extra} / N_{key}$

   Where N_{extra} - number of responses judged correct but not found.

4.2. Method

Each model element like class, relationship, attribute, packages, objects etc in the system response is compared with each element in the key & can be classified in three categories;

Correct- if it matches with an element in the answer key.

Incorrect- if it does not match with an element in the key.

Extra - if it is validation information from the text but is not in the key.

Here match is done in two ways. First, names of matching elements must be plausibly close (item can match with loan item but not with library). Second, an element specific process is carried out to see if the context of the element gives evidence that it really what its name suggests. Similarly other factors of models like attributes, relationships are matched both by approximate name matching & by context correctness.

Also to test the accuracy of the diagrams generated by the designed system, four parameters had been decided. Each generated class diagram is tested under: no. of objects and classes, no. of attributes, number of methods, number of associations and diagram labeling. Maximum score was declared 10. According to the wrong nominations and extractions, the points were counted. Comparing with existing tools, proposed systems performance can be shown below in Table 1.

Table 1. Performance evaluation based on precision, recall values

Sr. No	Tools	Recall Value	Precision Value
1	CM-Builder	73.00%	66.00%
2	GOOAL	-	78.00%
3	NL-OOML	-	82.00%
4	LIDA	71.32%	63.17%
5	Proposed System(URDA)	90%	90%

A matrix of results of generated diagrams is shown below in table 2. Which shows that there are very few tools those can extract information such as multiplicity, aggregations, generalizations, instances from natural language requirement. Thus, the results of this initial performance evaluation are very encouraging and support proposed approach and the potential of this technology in general.

Table 2: Comparison between UDRA & existing tools

Diagram Type	NL-OOPS	CM-Builder	LIDA	GOOAL	UDRA
Class diagram	NO	YES	YES	YES	YES
Object diagram	YES	NO	YES	NO	YES
Deployment diagram	NO	NO	NO	NO	YES
Package diagram	NO	NO	NO	NO	YES
Defining Operation	NO	NO	USER	YES	YES
Advance Relationship	NO	NO	USER	NO	YES

5. Result

We have tested this UDRA system for some case studies; following section includes experimental review of these.

Here we discuss case study from the domain of Library Information system. The goal of this case study is to test &

validate our approach. The problem statement for this case study is as follows:

"The Library system is used by informatics students & faculty. The Library contains books. Library contains journals. Books can be issued to students. Books can be issued to faculty. Faculty can also get journals from Library on their library card. Books can be issued only by Librarian. Librarian has name. Librarian gets salary. Journals can also be issued only be Librarian. The Library has Accountant. Accountant is responsible for receiving fine. Fine is charged only to students."

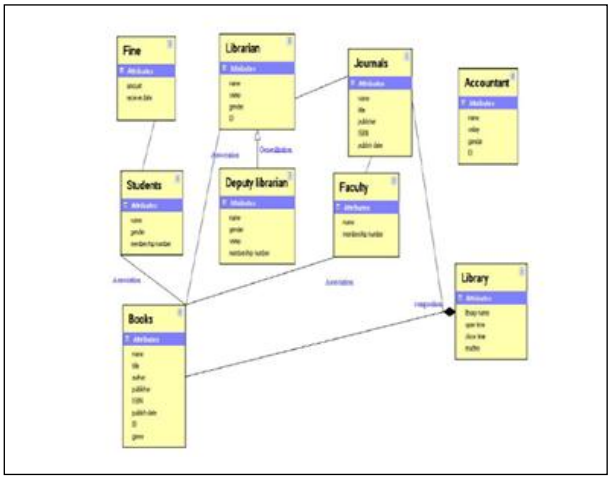

Fig 2 Class diagram generated by RACE system

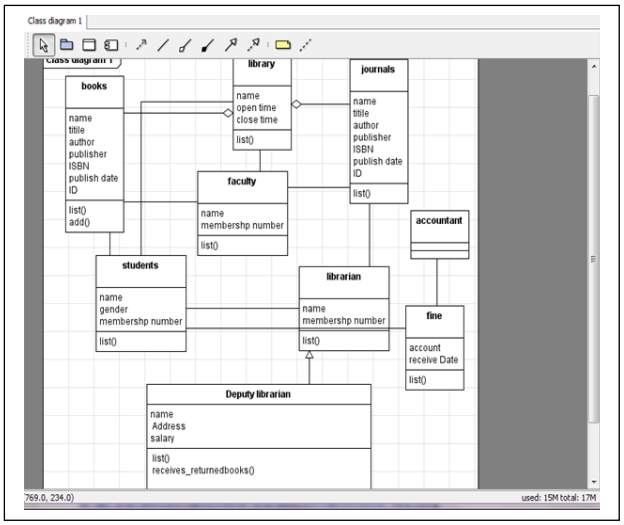

Fig 3Class diagram generated by UDRA system for Library

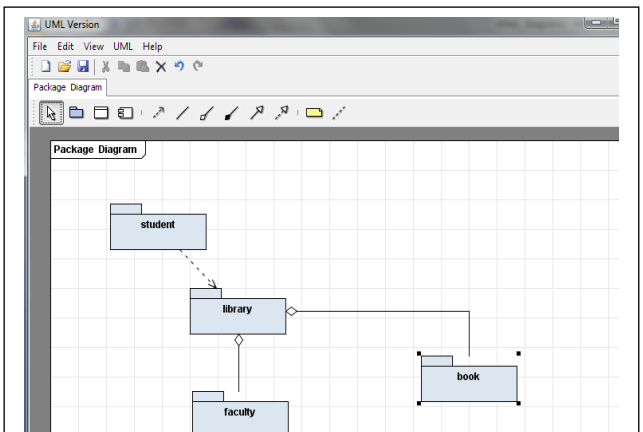

Fig 3Package diagram generated by UDRA system for Library

6. Conclusion

Our proposed method demonstrates the use of NLP and domain ontology techniques for reflecting natural language text in to of the UML diagrams. The designed system can generates UML diagrams like Class diagrams, Object diagrams, Package diagrams & Deployment diagrams. It generates complete class diagram including attribute specification, operations specifications as well as all advanced relationships like generalization, association, compositions etc. While using UDRA tool it is not mandatory for users to install diagram visualizing tool, it can be easily handled by UDRA built in system & it's GUI. The GUI provided by UDRA is user friendly that allows user to modify, print or edit generated diagrams. The concepts extracted UDRA are completely valid and recognized in the application domain.

7. References

1. [UML]Object Management Group. Unified modeling language specification (2.1.1). Tech-nical report, Frammingam, Mass, 2007. http://www.uml.org.
2. L. Goldin, and D. Berry, "AbstFinder, A Prototype Natural LanguageText Abstraction Finder for use in Requirement Elicitation,"Automated Software Engineering Journal, Vol 4, 2007,pp 73-80.
3. GobindaG.Chowdhury.(2001). Natural Language Processing.
4. R. J. Abbot, "Program Design by Informal English Description," ACM Journal, vol. 26, 1983, pp. 882-994
5. H. Buchholz, A. Dusterhoft, B. Thalheim, "Capturing Information on Behavior with the RADD_NLI: A Linguistic and Knowledge Base Approach," in Proceedings Second Workshop Application of Natural Language to Information System, IOS Press, 1996, pp. 185–196.
6. S. Naduri, S. Rugaser, "Requirements Validation through AutomatedNatural Language Processing," in Proceedings twenty eighth HawaiiInternational Conference Systems Science, 1995, pp.362–367.
7. G. Hector Perez Gonsalves and K. JugalKalita , "Automatically Generating Object Models from Natural Language Analysis," in Proceedings seventeenth annual ACM SIGPLAN conference on Object-oriented Programming systems, languages, and applications, Seattle, Washington, 2002, pp. 86–87
8. K. Li, "Towards Semi-automation in Requirements Elicitation : mapping natural language and object oriented concepts," inProceedings thirteenth IEEE International Requirements Engineering Conference, 2005
9. M. Saeki, H. Horai, H. Enomoto, "Software Development Process from Natural Language Specification", in Proceedings eleventh International Conference on Software Engineering, 1989, pp 64-73
10. Overnyer, B. Lavoie and O. Rambow, "Conceptual Modeling through Linguistic Analysis using LIDA", in Proceedings twenty third International Conference on Software Engineering, 2001, pp. 401-410.
11. M. Brambilla, S. Ceri, S. Comai, P. Fraternali, "A CASE tool for modeling and automatically generating web-service enabled applications," in International Journal Web Engineering and Technology, vol. 2, Number 4, 2006, pp 354-372
12. Hjkhjkhjk…A. Oliveira, N. Seco and P. Gomes, "A CBR Approach to Text to Class Diagram Translation", in TCBR

Workshop eighth European Conference on Case-Based Reasoning, Turkey, September, 2006.
13. L. Mich, R.Garigliano, "A linguistic approach to the development of object-oriented system using the NL system LOLITA", Object Oriented Methodologies and Systems, (ISOOMS), LNCS 858, pp 371-386.
14. P. Sawyer, Rayson, Garside, "REVERE: support for requiremnt sysnthesis from documents", in Information systems Frontier Journal, Vol 4, 2002, pp. 371-386.
15. Scott P. Overmyer, Benoit Lavoie, Owen Rambow, "Conceptual Modeling through Linguistic Analysis Using LIDA", Proceedings of the 23rd International Conference on Software Engineering, ICSE, pp. 401-410, May 2001
16. H. M. Harmain and R. Gaizauskas, "CM-Builder: A Natural Language-based CASE Tool", Journal of Automated Software Engineering, pp. 157-181, 2003.
17. KalaivaniSubramaniam, Dong Liu, Behrouz H. Far and Armin Eberlein Department of Electrical and Computer Engineering, "UCDA: Use Case Driven Development Assistant Tool for Class Model Generation", University of Calgary 2500, University Drive, N.W., Calgary, Alberta, Canada, T2N 1N4, 2004.
18. Vincenzo Ambriola, Vincenzo Gervasi, "On the Systematic Analysis of Natural Language Requirements with CIRCE" in Springer Science (Dipartimento di InformaticaUniversit`a di Pisa, Italy), 2006.
19. Manuel Clavel, Marina Egea, Viviane Torres da Silva, "The MOVA Tool: A Rewriting-Based UML Modeling, Measuring, and Validation Tool", in Proc. 12th Conference on Software Engineering and Databases Zaragoza (Spain), 2007
20. D. Deva Kumar and R. Sanyal, "Static UML Model Generator from Analysis of Requirements (SUGAR)," Proc. Advanced Software Engineering and Its Applications, ASEA 2008, pp. 77-84, 2008.
21. Hema Krishnan, "Relative Extraction Methodology for Class Diagram Generation using Dependency Graph", Department of Computer Science Cochin University of Science, 2008.
22. Deva Kumar Deeptimahanti, Muhammad Ali Babar Lero, University of Limerick, Ireland. "An Automated Tool for Generating UML Models from Natural Language Requirements (UMGAR)", IEEE/ACM International Conference on Automated Software Engineering, 2009.
23. Imran SarwarBajwa, Ali Samad, ShahzadMumtaz," Object Oriented Software Modeling Using NLP Based Knowledge Extraction", European Journal of Scientific Research ISSN 1450-216X Vol.35 No.1, pp 22-33, 2009
24. Ibrahim , Mohd and Ahmad Rodina , " Class Diagram Extraction from Texual Requirements Using Natural Language Processing(NLP) Techniques " In of Proceedings of the 2010 second.
25. D. Rusu, "Triplet Extraction from sentences," in Proceedings tenth International Multi Conference, Information Society, 2007.
26. 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.
27. G. Hector Perez Gonsalves and K. JugalKalita, "Automatically Generating Object Models from Natural Language Analysis," in Proceedings seventeenth annual ACM SIGPLAN conference on Object-oriented Programming systems, languages, and applications, Seattle, Washington, pp. 86–87, 2002.
28. WordNet(2.1)http://www.cogsci.princeton.eu/~wn/.
29. Xiaohua Zhou and Nan Zhou, "Auto-generation of Class Diagram from Free-text Functional Specifications and Domain Ontology", 2004.
30. Rebecca Wirfs-Brock, "Object Design: Roles, Responsibilities, and Collaborations", Addisson-Wesley, 2002.
31. G. Booch, "Object Oriented Design with Applications", the Benjamin Cummings Publishing Company, 1991
32. Gobinda G. Chowdhury, "Natural Language Processing", Johanson Publishing company, 1998.
33. FaridMeziane, Nikos Athanasakis, Sophia Ananiadou, "Generating Natural Language specifications from UML class diagrams", Springer-Verlag London Limited pp.46-52, 2007.
34. Haruhiko Kaiya, Motoshi Saeki, "Ontology Based Requirements Analysis: Lightweight Semantic Processing Approach", Proceedings of the Fifth International Conference on Quality Software (QSIC'05), 2005 IEEE.
35. K. Li, R. G. Dewar, R. J Pooley, "Object Oriented Analysis Using Natural Language Processing", in Proceedings Saudi International Innovation Conference, pp 87-95, Automated Software Engineering Journal, Vol 4, pp 73-80, 2007.
36. Li Tan, Zongyuan Yang and JinkuiXie, "OCL Constraints Automatic Generation for UML Class Diagram", East China Normal University Shanghai, China IEEE, pp.173-179,2010.