ISSN: 2277-3754
ISO 9001:2008 Certified
International Journal of Engineering and Innovative Technology (IJEIT)
Volume 1, Issue 6, June 2012

Semantics Based Identifier Mining for UML Models

Bala Sundara Ganapathy Nadesan, Dr.K.Alagarsamy

Abstract— In order to develop quality software, it must be designed according to the requirements. Software requirements have a brunt on the design, and the design, in turn, has a severe brunt on the software development phase. UML diagram is an ideal choice for software developers who need to demonstrate and deduce relationships, actions, and connections of a software application using the Unified Modeling Language (UML) notation. It has been the design tool of choice for years with the supremacy of object oriented software engineering. The software designer must go through the software requirements specifications (SRS) and has to select the identifiers manually for the UML Models. The proper choice of identifiers facilitates software understandability and maintainability. Several software engineers and researchers have stated that identifiers are the most vital informative components of software development entities. The proposed approach extracts the valid information from the software requirement specification and suggests suitable names for identifiers for constructing UML Models. This could be achieved by using a semantic analyzer that recognizes the given identifiers and extracts relevant short identifiers using dictionary. If the identifier name is large, then suitable shorter name can be extracted using semantic analyzer.

Index Terms—**Identifier, Software Requirements Specification, Semantic Analyzer, Unified Modeling Language.**

I. INTRODUCTION

Even though the phrase ‘software crisis’ is 40 years old, software still takes too long to develop, costs too much, and does not work well when eventually delivered. System design is one of the most decisive and volatile phase in the software engineering process. The software development and the process that brings software into work out everyday activities have become a critical issue for modern organizations. The traditional way of software development has always been questioned, and people have been looking for new ways to improve the software development process. Most of the software companies develop the software by a group of people, even 1,000 to 10,000 members. These members are divided into some groups like software analysts, software designers, software developers, software testers, and software managers. Software analyst has to gather the software requirements specification from the client. After proper study of system document as a whole, they may prepare software requirements specification and it has to be duly approved by the client. The designer has to go through the SRS document completely and then go for designing the software [9]. Numerous approaches and methods intended at highlighting software design problems and at sustaining designers in improving software quality have been proposed in recent years [1], [2]. One of the major tools used for designing the software is Unified Modeling Language. Using UML models one can easily describe the system. Large design models contain thousands of model elements. Software Engineers easily get overwhelmed maintaining the reliability of such design models over time [3], [4] and choosing reliable identifier for their UML Model is indispensable.

II. PROBLEM DEFINITION

A. Need of Software Requirement Specification

A software requirements specification (SRS) is a complete description of the activities of the software to be developed. It comprises a set of use cases that depict all of the interactions that the users will have with the software. SRS also includes functional requirements, which define the internal workings of the software like technical details, data manipulation and processing, calculations, and other specific functionalities. In addition, it also contains nonfunctional requirements, which oblige conditions on the system design or implementation. Therefore SRS is basic core document that is needed for the software development process. The proposed system extracts the valid information from the SRS and suggests suitable name for identifiers for constructing UML Models.

B. Need of UML diagrams

Software is very intangible and hard to visualize. A visual modeling language, such as Unified Modeling Language (UML), allows software to be developed visualized in multiple dimensions, so that the software engineers could completely understand before construction begins [10], [11]. The overall possibility of the software can quickly and easily be defined at the beginning of the software development with a high level model allowing for precise estimation. Additional detail can then be added to each part of the software as it is constructed, until finally the system emerges as code [9], [14], [17]. UML has two diagrams that are used for behavior specification: the activity diagram and the state diagram. These two diagrams and the class diagram which is used for modeling the structural aspects of the object model give the framework that allows 100% code generation [12], [13], [18]. Therefore if we choose identifiers for the construction of UML models, we don’t need to bother about selecting proper and relevant identifiers for the source code. If UML model has the relevant identifier, then its converted source code also will have the same set of identifiers.

ISSN: 2277-3754
ISO 9001:2008 Certified
International Journal of Engineering and Innovative Technology (IJEIT)
Volume 1, Issue 6, June 2012

C. Need of proper Identifiers

Quality is important to all software engineering projects as it affects the bottom line: lower quality leads to higher costs [16]. Software quality is impacted by program, and program quality is impacted by design entities such as UML Model and their identifier naming [6], [7]. The particular interest herein is the key role that identifier naming plays in the design quality. There are various earlier studies that motivated us for studying identifiers. Deissenbock and Pizka state that "research on the cognitive processes of language and text understanding shows that it is the semantics inherent to words that determine the comprehension process" [8]. Another motivation for studying identifiers comes from Rilling and Klemola who note that "In computer programs, identifiers represent defined concepts" [15]. Caprile and Tonella state that "Identifier names are one of the most important sources of information about program entities" [5]. Takang et al. opine that "The quality of identifier names is an issue that merits closer consideration and exploration in its own right" [19]. Therefore the proper choice of identifier for the UML models is obligatory.

III. A NARRATIVE APPROACH FOR IMPROVING THE QUALITY OF THE IDENTIFIER MATH

This section describes a narrative approach for improving the quality of the identifiers used in various UML Models during software development. The proposed approach is based on the assumption that system designers are induced to make the UML Models and its identifiers more consistent with domain terms if the software development environment provides information about the textual similarity between the UML model being drawn and the related high-level artifacts [21], [22] . Clearly, the proposed approach is based on the assumption that high-level documentation like System Requirement Specification (SRS) and module specification is available during the development process. Figure.1 shows the flow of information between a designer and the Integrated Development Environment (IDE) in the proposed approach.

A. Importing Software Requirement Specification

The proposed system has to accept software requirement specification document either available as a text file or word processing document, from which terms are extracted [27] and filtered for identifying the suitable identifiers for the construction of UML models.

B.Term Extraction and Filtering

When designers are designing a UML model artifact, they can be continuously informed about the list of identifiers needed for the model. To do this, list of identifiers has been extracted and filtered from the system requirement specifications or model specifications. In order to extract and filter the relevant identifiers, the system will find the textual tokens and punctuations.

e.g.,

Sample System requirement Specification: Vendor Master

- Vendor Master details are maintained by the Purchase section along with their information that are specified below (Societies & Suppliers of Yarn, Silk and Cotton)

- Basic details (General Information) like Vendor Id, Vendor Name, Primary Address, Contact Address, and Telephone Numbers.

- Account details of the vendor like the mode of payment, currency used for payment, vendor account to be credited, vendor bank name and account number

- Transaction history of the vendor which includes the monthly transaction details for the electricity.

From the above specification system will find the textual token Vendor Master and its module specific identifiers like Vendor Id, Vendor Name, Primary Address, Contact Address, Telephone Numbers, Email Id etc. that could be extracted using delimiters [23].

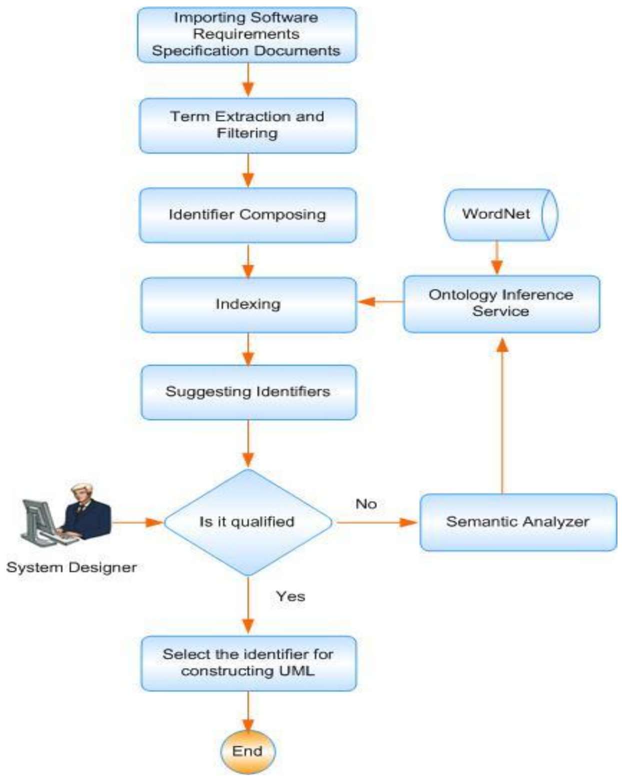
Importing Software Requirements Specification Documents
Term Extraction and Filtering
Identifier Composing
WordNet
Indexing
Ontology Inference Service
Suggesting Identifiers
System Designer
Is it qualified
No
Semantic Analyzer
Yes
Select the identifier for constructing UML
End

Fig 1. Improving the Quality of the Identifier

C. Identifier Composing

The lists of identifiers are composed from the specification document. These high-level artifacts are then indexed using any suitable data structures, and it has been suggested for the construction of UML models. The system designer has to select the identifiers for their models. If the suggested identifiers are too long or not relevant, then query has to be sent to the semantic analyzer for its betterment [24], [26]. Semantic analyzer extracts relevant identifiers using Ontology Inference Service. The OIS query results are indexed and then again sent to the system designer for the identifier selection.

D. Ontology Construction

Ontology Construction is being built using WordNet, a linguistic ontology. WordNet is a semantic lexicon for the English language. It groups English words into sets of synonyms called synsets; provides short, general definitions; and records the various semantic relations between these synonym sets. The purpose is twofold: to produce a combination of dictionary and thesaurus that is more intuitively usable, and to support automatic text analysis and artificial intelligence applications. The database and software tools can be downloaded and used freely. The database can also be browsed online. WordNet was created and is being maintained at the Cognitive Science Laboratory of Princeton University under the direction of psychology professor George A. Miller. Development began in 1985. Over the years, the project received about $3 million of funding, mainly from government agencies interested in machine translation**.** As of 2006, the database contains about 150,000 words organized in over 115,000 synsets for a total of 207,000 word-sense pairs; in compressed form, it is about 12 megabytes in size. Thus with the help of this ontology, an investigator searching for a term will be able to discover. The use of ontology will allow the search capability on the metadata catalog and other web resources beyond just using keywords.

E. Ontology Inference Service

The Ontology Inference Service (OIS) is a Java API for WordNet Searching (JAWS) that provides Java applications with the ability to retrieve data from the WordNet database [28], [29]. It is a simple and fast API that is compatible with both the 2.1 and 3.0 versions of the WordNet database files and can be used with Java 1.4 and later. JAWS were created by Brett Spell as a project for Dr. Margaret Dunham's class on Information Retrieval at Southern Methodist University .

IV. CONCLUSION

The paper proposed a narrative approach to help designers to improve the quality of the identifiers used in the UML model. Constructing research artifact and assessing the same to test for quality, effectiveness, and efficiency, and abstract the knowledge gained in terms of design ethics and theories are among the important research activities in design science research. It's future work to enhance the process of extracting the suitable identifiers from the SRS, and bring out its implementation to carrying out explorations of classes of identifier mining tasks. Experimentation and evaluation of the above suggested aims to find the quality of design model that leads to the quality of the software could be undertaken.

REFERENCES

[1] Alexander Egyed, "Automatically Detecting and Tracking Inconsistencies in Software Design Models" IEEE Trans. Software Eng., vol. 37, no. 2,pp. 188 – 204, Mar/Apr. 2011.

[2] S. Kim, E.J. Whitehead, Jr., and Y. Zhang, "Classifying Software Changes: Clean or Buggy?" IEEE Trans. Software Eng., vol. 34, no. 2, pp. 181-196, Mar./Apr. 2008.

[3] S. Kim, T. Zimmermann, E.J. Whitehead Jr., and A. Zeller, "Predicting Faults from Cached History," Proc. 29th Int'l Conf. Software Eng., pp. 489-498, 2007.

[4] D. Lawrie, H. Feild, and D. Binkley, "Quantifying Identifier Quality: An Analysis of Trends," Empirical Software Eng., vol. 12, no. 4, pp. 359-388, 2007.

[5] B. Caprile and P. Tonella. Restructuring program identifier names. In ICSM, 2000.

[6] D. Lawrie, C. Morrell, H. Feild, and D. Binkley, "Effective Identifier Names for Comprehension and Memory," Innovations in Systems and Software Eng., vol. 3, no. 4, pp. 303-318, 2007.

[7] D. Lawrie, C. Morrell, H. Feild, and D. Binkley, "What's in a Name? A Study of Identifiers," Proc. 14th IEEE Int'l Conf. Program Comprehension, pp. 3-12, 2006.

[8] F. Deissenbock and M. Pizka. Concise and consistent naming. In Proceedings of the 13th International Workshop on Program Comprehension (IWPC 2005), St. Louis, MO, USA, May 2005. IEEE Computer Society.

[9] A. Abadi, M. Nisenson, and Y. Simionovici, "A Traceability Technique for Specifications," Proc. 16th IEEE Int'l Conf. Program Comprehension, pp. 103-112, 2008.

[10] G. Antoniol, G. Canfora, G. Casazza, A. De Lucia, and E. Merlo, "Recovering Traceability Links between Code and Documentation," IEEE Trans. Software Eng., vol. 28, no. 10, pp. 970-983, Oct. 2002.

[11] L.C. Briand, Y. Labiche, and L. O'Sullivan, "Impact Analysis and Change Management of UML Models," Proc. Int'l Conf. Software Maintenance, p. 256, 2003.

[12] L.A. Campbell, B.H.C. Cheng, W.E. McUmber, and K. Stirewalt, "Automatically Detecting and Visualizing Errors in UML Diagrams," Requirements Eng. J., vol. 7, pp. 264-287, 2002.

[13] A. Egyed, "Automated Abstraction of Class Diagrams," ACM Trans. Software Eng. and Methodology, vol. 11, pp. 449-491, 2002.

[14] A. Egyed, "Instant Consistency Checking for the UML," Proc. 28th Int'l Conf. Software Eng., pp. 381-390, 2006.

[15] J. Rilling and T. Klemola. Identifying comprehension bottlenecks using program slicing and cognitive complexity metrics. In Proceedings of the 11th IEEE International Workshop on Program Comprehension, Portland, Oregon, USA, May 2003.

[16] H. Saiedan and L. M. Mc Clanahan. Frameworks for quality software process: SEI capability maturity model. Software Quality Journal, 5(1):1, 1996.

[17] J. Rumbaugh, J. Ivar, and B. Grady, The Unified Modeling Language Reference Manual. Addison Wesley, 1999.

[18] G. Antoniol, G. Casazza, and A. Cimitile, "Traceability Recovery by Modeling Programmer Behavior," Proc. Seventh Working Conf. Reverse Eng., pp. 240-247, 2000.

[19] A. Takang, P. Grubb, and R.Macredie. The effects of comments and identifier names on program comprehensibility: an experiential study. Journal of Program Languages, 4(3), 1996.

[20] A. De Lucia, F. Fasano, R. Oliveto, and G. Tortora, "Recovering Traceability Links in Software Artifact Management Systems Using Information Retrieval Methods," ACM Trans. Software Eng. and Methodology, vol. 16, no. 4, 2007.

[21] A. Marcus and J.I. Maletic, "Recovering Documentation-to-Source-Code Traceability Links Using Latent Semantic Indexing," Proc. 25th Int'l Conf. Software Eng., pp. 125-135, 2003.

[22] R. Settimi, J. Cleland-Huang, O. Ben Khadra, J. Mody, W. Lukasik, and C. De Palma, "Supporting Software Evolution through Dynamically Retrieving Traces to UML Artifacts," Proc. Seventh IEEE Int'l Workshop Principles of Software Evolution, pp. 49-54, 2004.

[23] N. Anquetil and T. Lethbridge, "Assessing the Relevance of Identifier Names in a Legacy Software System," Proc. 1998 Conf. Centre for Advanced Studies on Collaborative Research, 1998.

[24] F. Deissenboeck and M. Pizka, "Concise and Consistent Naming," Software Quality J., vol. 14, no. 3, pp. 261-282, 2006.

[25] D. Lawrie, H. Feild, and D. Binkley, "An Empirical Study of Rules for Well-Formed Identifiers," J. Software Maintenance, vol. 19, no. 4, pp. 205-229, 2007.

[26] B. Caprile and P. Tonella, "Nomen Est Omen: Analyzing the Language of Function Identifiers," Proc. Sixth IEEE Working Conf. Reverse Eng., pp. 112-122, 1999.

[27] D. Lawrie, H. Feild, and D. Binkley, "Extracting Meaning from Abbreviated Identifiers," Proc. Seventh IEEE Int'l Working Conf. Source Code Analysis and Manipulation, pp. 213-222, 2007.

[28] John Davies, Dieter Fensel, and Frank Van Harmelen, **"**Towards the Semantic Web: Ontology-driven Knowledge Management," J. Wiley, 2003.

[29] Latifur Khan, "Ontology-based Information Selection," Ph.D. Thesis, University of South California, 2000.

AUTHOR'S PROFILE

Bala Sundara Ganapathy Nadesan received the Master's Degree in Computer Applications in 2001 and the Master of Philosophy Degree in Computer Science in 2007 from Madurai Kamaraj University, India. He has registered for Ph.D. in the same University and is currently pursuing research in software engineering. He is working as an Assistant Professor at Panimalar Engineering College, Chennai, India. He has published research articles in international journals and conference proceedings, and authored textbooks. He serves as an editorial and reviewer board member of International Conference on Information Science and Applications.

Dr. K.Alagarsamy is the Associate Professor at Madurai Kamaraj University, India. He secured his M.C.A., M.Phil. and Ph.D. in Computer Science from Madurai Kamaraj University, India. He has published 18 international research articles and few books, and participated in many international seminars. He has 28 years of experience in teaching and research. His current research includes software engineering, program comprehension, reverse engineering, reengineering, software configuration management, workflow management, document management, visual languages, web engineering, e-learning and data mining.