Review on Reverse Engineering Techniques of Software Engineering

Upasana Choudhary
Student (M.Tech)
Department of computer science
Sanghvi Institute of Management & Science, Indore

Maya Yadav
Assistant professor
Department of computer science
Sanghvi Institute of Management & Science, Indore

ABSTRACT
Reverse Engineering is an approach to extract requirement information application from XML at higher level of abstraction. In the present study XML to UML transformation methods has been explored and discussed along with some other related work in reverse engineering. A brief review of reverse engineering shows that the transformation results of XML to UML are beneficial for developer. However it does not show changes as per the requirements view. Generation of Natural Language specifications from UML class diagrams is also discussed where results were found encouraging and feasible. All related work of reverse engineering of XML and Natural language specification from UML is clear but a lot of work is to be done to solve real world problems satisfactorily. Capability of different reverse engineering tools is also discussed. Automatic code generation using UML to XML schema transformation is reviewed which shows that it can reduce the time and efforts of coding.

Keywords
XML, Reverse Engineering, Conceptual Model

1. INTRODUCTION

Reverse engineering is the process of extracting knowledge and the requirement specification of a product from an analysis of its code [8]. The reasons and goals for obtaining such information vary widely from every day or socially beneficial actions, to criminal actions, depending upon the situation.

2. BACKGROUND

Extensible Markup Language is a text-based format that allows for the structuring of electronic documents and is not limited to a set of labels. Extensible Markup Language is used to describe data. The XML standard is a flexible way to create information formats and electronically share structured data via the public Internet, as well as via corporatenetworks.XML code, a formal recommendation from the World Wide Web Consortium is similar to Hypertext Markup Language. Both XML and HTML contain markup symbols to describe page or file contents.

XML is a technology used in a wide variety of scenarios, from a message format used by web services to storage of data in databases. As the number of possible usages of XML grows, so does the need of easy management of large numbers of XML data sources and their integration. There we can use conceptual modeling of XML data. It allows a domain expert to model the problem domain independently of the implementation and then create corresponding XML schemas, which are used to describe a structure of XML documents [2].

SRS is a complete description of the behavior of a system to be developed and may include a set of use cases that describe interactions the users will have with the software. In addition it also contains non-functional requirements. Non-functional requirements impose constraints on the design or implementation.

3. MOTIVATION

The important reason focusing in reverse engineering, as if the process needed to recheck XML schema developed with the earlier UML diagram created [1,2,3,4]. No UML diagram documented, if the developer has not gone through formal development phase. In both situations, requirement documentation is very important to tell the progress of project development in the track.

4. SURVEY

Object-oriented modeling languages began appearing as early as the mid-1970s when methodologists experimented with various approaches to object-oriented analysis and design [2]. During the period from 1989 – 1994, the number of identified modeling languages increased from less than 10 to more than 50. However, object methods users had trouble finding any one modeling language that they were completely satisfied with. It became apparent that a standard modeling language was needed.

To tackle this problem, Rational Software and a few other organizations joined forces to begin development on the Unified Modeling Language (UML) in 1994, and UML was adopted by the Object Management Group, Inc. (OMG) in 1997. Since then the OMG has assumed responsibility for the further development of the UML standard [2].

Extensible Markup Language (XML) has become a standard for data representation and exchange over the Internet. XML schemas are often used to define vocabularies of XML document types and to validate whether the XML documents adhere to the rules defined in the XML schemas [4]. Since XML schemas are textual, programmatic, logical-level schemas, users of XML schemas often find it difficult to understand and communicate with each other the structure and content of the XML schemas and documents as the XML schemas grow in complexity.

Anshul et al. [1] has proposed Automatic Code Generation Using Uml To Xml Schema Transformation. With the help of UML class diagram XML schema was generated and then this

XML schema was used for code generation. Authors have implemented a system for XML Schema generation using .net platform. Evaluation results shows that this automatic code generation can reduce efforts and development time by reducing coding efforts.

Augustin Yu and Robert Steele [2], An Overview of Research on Reverse Engineering XML Schemas into UML Diagrams has shown that a solution to the above problem would be to convert the logical-level XML schemas developed back to conceptual-level Unified Modeling Language diagrams to facilitate easy understanding and communication. They have described an overview of research on reverse engineering XML schemas into UML diagrams. Conclusion given is that the reverse engineering of XML schemas to UML diagrams. Of the existing research done so far on the reverse engineering approaches, different methods each has its pros and cons, and a standard optimal approach has yet to be reached and agreed on.

Christof Lutteroth [3] has presented an Automated Reverse Engineering of Hard-Coded GUI Layouts which is able to recover a higher-level layout representation of a hard- coded GUI using the Auckland Layout Model, which is based on the mathematical notion of linear programming. Study evaluated that reverse engineering of hard-coded GUIs into higher-level specifications is desirable, and that it can be performed successfully using the Auckland Layout Model. The presented algorithm can be used to achieve the following specific goals.

HannaniAman and Rosziati Ibrahim [4], Reverse Engineering: From Xml to Uml for Generation of Software Requirement Specification focused on XML to UML conversion and has proposed method of reverse engineering from XML to generate Software Requirement Specification in a documented manner. Through this research document above author shows the framework of reverse engineering from XML to UML for generating software requirement specification. Since the transformation XML to UML has undergone an intensive research, extending the XML to SRS will be beneficial to project development activity in modern methodologies for saves time and development cost. This framework for reversing from XML to generate Software Requirement Specification has been established.

Jakub Klimek, Martin Necasky [5], On Inheritance in Conceptual Modeling for XML, analyzed that manual change management of the XML formats may be error-prone and time consuming. Through their research they tackled this problem in their previous work with the introduction of a formal two level conceptual model for XML which interconnects multiple XML schemas describing parts of a common problem domain on a conceptual level. This allows for well-defined and automated change management of XML schemas. Hence they extended previous work with inheritance modeling. Because inheritance is common in XML schemas and conceptual models in general, showed how inheritance can be modeled on a platform-independent and platform-specific levels and also showed how the constructs can be translated to the XML Schema language. Two basic types of inheritance modeling: structural and conceptual inheritance shown in Figure 1. The differences and how these two types need to be reflected in their model is discussed.

Jakub Klimek and Martin Necasky [6], Reverse-engineering of XML Schemas:A Survey, have been proposed Some methods for reverse-engineering of XML schemas, they are compared using various criteria such as used XML schema languages, level of user involvement, number of XML schemas that can be covered by the conceptual model or support for consecutive XML schema evolution. They are also evaluated according to their potential to be used as parts of a system for management, evolution and integration of XML as a whole. On comparison and evaluation of several approaches for reverse engineering of XML schemas according to given comparison criteria. Among them, only one was well suited for being a part of a larger system for evolution i.e. “Reverse Engineering of XML Schemas to Conceptual Diagrams” by Martin Necasky.

Since one of the prominent characteristics of XML applications is their dynamic nature. Changes in user requirements cause changes in schemas used in the systems and changes in the schemas subsequently make existing documents invalid. In this work,

Jakub Maly, Martin Necasky and IrenaMlynkova [7], Efficient adaptation of XML data using a conceptual model, studied two tightly coupled problems—schema evolution and document adaptation. Their approach extends an existing conceptual model for evolution of XML applications towards document adaptation, by introducing a formal framework for detecting changes between two versions of a schema. From the detected changes it was possible to create a script that transforms documents valid against the old version of the schema to documents valid against its new version.

Mamta garg and manoj kumar jindal [8], have discussed software reverse engineering and hardware reverse engineering along with various reverse engineering tools which help in developing software design better by the use of existing source code. Also discussed that how Reverse engineering of software can make use of the clean room design technique to elude violating copyrights. The Discussion of various tools of reverse engineering i.e. Hexadecimal Dumprer, Debugger, Fault Injection Tools, The Disassembler, Compiler etc. brought to a conclusion that reverse engineering of any product uncovers as much information as possible about the design ideas that were used to produce a particular product.

Shivani Budhkar and arpita gopal [9], presented An Experience Report on Reverse Engineering Java Code to Class Diagram. The study focused on analyzing the potential of four software reverse engineering tools (i.e. Rational rose, ArgoUM, Reverse, Enterprise Architecture (EA)) to create class diagram from java source code. Study shows that however all tools provide reverse engineering facility; most of them are able to recognize basic features like classes and associations. ArgoUML could not show that. Several advanced concepts like multiplicity provided that inverse association etc have not been addressed in these tools.

FaridMeziane, Nikos Athanasakis and Sophia Ananiadou [10], Generating Natural Language specifications from UML class diagrams, proposed a system that generates Natural Language specifications from UML class diagrams to improve the process of software development process. At first they investigated the variation of the input language used in naming the components of a class diagram based on the study of a large number of examples from the literature and then developed rules for removing ambiguities in the subset of Natural Language used within UML. They used WordNet, a linguistic ontology, to disambiguate the lexical structures of the UML string names and generate semantically sound sentences. Their system is developed in Java and is tested on an independent though academic case study. Process adopted for doing this work is shown below Figure 2.

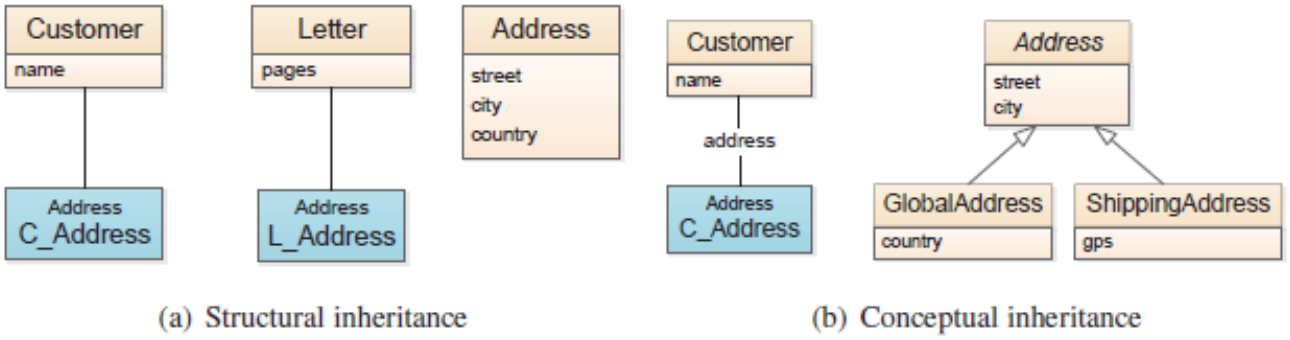

Figure 1.Two inheritance types in PSM schemas [5]

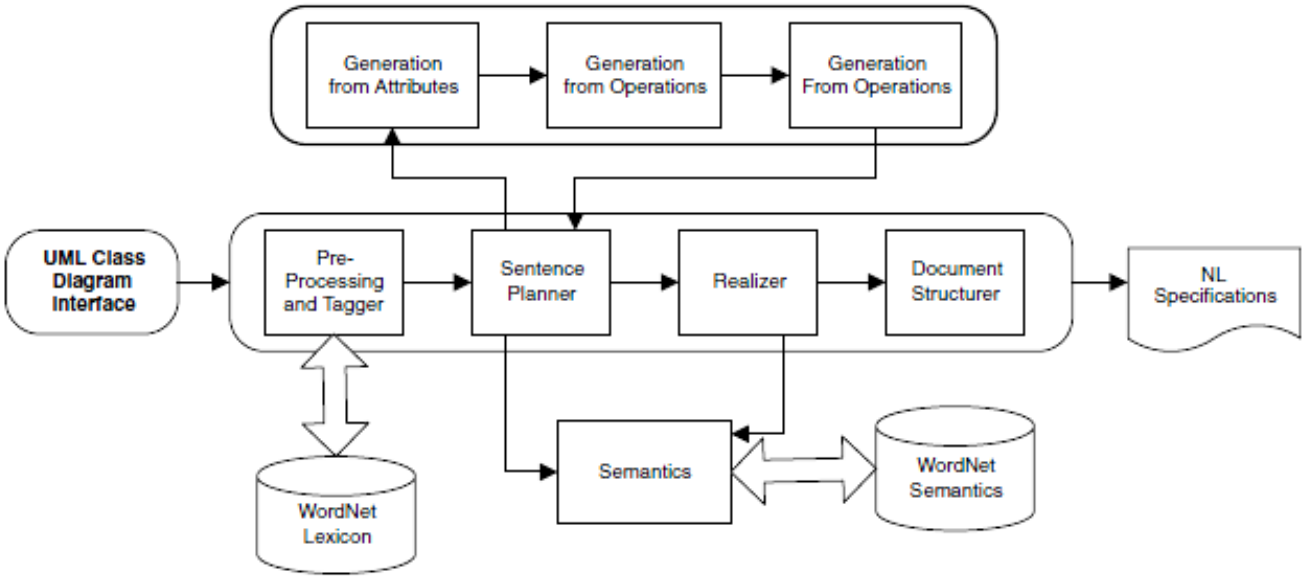

Figure 2.The GeNLang UML System's Architecture [10]

5. CONCLUSION

In depth review of Reverse Engineering shows that, so many researchers are putting their efforts to ease the software development process through various methods. The conclusion can be made that all the methods discussed in this paper are feasible but has never been developed to the requirement level. Every method has their pros and cons. Automatic code generation can reduce efforts and development time by reducing coding efforts.

Future works can be extended towards better implementation of reverse engineering methods proposed in various papers [5, 6 & 7]. New research can be focused on some more automatic code generation methods [1 & 3].

Every researcher whoever is working in this field of software engineering, for them implementation of all of this methods to solve any real world problem should be the top most priority. Or otherwise theoretical concepts will not be worthwhile.

6. ACKNOWLEDGMENTS

We are very thankful to all the experts who have contributed towards development of the manuscript.

7. REFERENCES

[1] Anshul, Sompal, Vikas Sheoran, 2014. Automatic Code Generation Using Uml to Xml Schema Transformation. International Journal of Advancement in Engineering Technology, Management & Applied Science ISSN No:2349-3224

[2] Augustin Yu., Robert Steele, 2005. An Overview of Research on Reverse Engineering XML Schemas into UMLDiagrams. Proceedings of the Third International Conference on Information Technology and Applications (ICITA'05). IEEE Computer Society. 0-7695-2316-1

[3] Christof Lutteroth, 2008. Automated Reverse Engineering of Hard-Coded GUI Layouts. Conferences in Research and Practice in Information Technology, Vol. 76.

[4] Hannani Aman, Rosziati Ibrahim, 2013. Reverse Engineering: From Xml to Uml for Generation of Software Requirement Specification.8th International Conference on Information Technology in Asia (CITA).978-1-4799-1092-2

[5] Jakub Klimek, Martin Necasky, 2012. On Inheritance in Conceptual Modeling for XML. The 3rd International Conference on Ambient Systems, Networks and Technologies(ANT), ELSEVIER Procedia Computer Science 10 (2012) 54 – 61

[6] Jakub Klimek, Martin Necasky, 2010. Reverse-engineering of XML Schemas:A Survey. pp. 96{107, ISBN 978-80-7378-116-3

[7] Jakub Maly, Martin Necasky, Irena Mlynkova, 2014.Efficient adaptation of XML data using a conceptual model. Published online: 19 September 2012 Springer Science+Business Media, LLC 2012,InfSyst Front (2014) 16:663–696

[8] Mamta Garg , Manoj Kumar Jindal, 2009. Reverse Engineering – Roadmap to Effective software Design. International Journal of Recent Trends in Engineering, Vol. 1, No. 2(ACEEE)

[9] Shivani Budhkar, Dr. Arpita Gopal 2011. Reverse Engineering Java Code to Class Diagram: An Experience Report. International Journal of Computer Applications (0975 –8887) Volume 29– No.6

[10] FaridMeziane, NikosAthanasakis, Sophia Ananiadou, Generating Natural Language specifications from UML classDiagrams, Published online: 25 September 2007, Springer-Verlag London Limited 2007, DOI 10.1007/s00766-007-0054-0