Recovering UML Class Diagram from Java Source Code

Ye Ye Khaing, Pearl
Computer University (Pathein), Myanmar
yyekhaing @gmail.com, pearl417@gmail.com

Abstract

Recovering the static structure of legacy source code is important in program understanding. A complex system can be simplified by a model exposing only some aspects of the system. In this system, we apply one of the most modeling methodologies named unified modeling language (UML). It is a language for visualizing, specifying, constructing, and documenting the artifacts of a software intensive system. The mind finds it easier to work with a graphics than with a long list of complex instructions. This paper presents a simple but effective approach that convert Java code to Extensible Markup Language (XML), then to an UML diagram. Using this approach we intended to develop a CASE tool for reverse engineers by combining both Java and XML technologies. In this system XML document are used to store program information (parse result). This system provides to produce generalization and realization relationship only. So, we can easily rearrange, sort, or otherwise modify the information to gain the UML model more precisely.

Keywords: Reverse Engineering, UML, XML

1. Introduction

Today's corporate world is dynamic and software artifacts have to be adaptable to keep pace with business needs. Hence, there is a need for tools to automate the modification and development of existing system. To maintain the legacy software, code understanding is an essential procedure. However, this process can be time consuming and more difficult by a lack of good documentation. It is often the case that maintainers of codes are not the original designers [2]. So, programmers spend large amounts of time to understand other people's code. In fact, it is estimated that 50% of a software engineer's time is spent on maintenance tasks involving information searching and program understanding.

Analyst produce design models, which is the underlying program architecture and behavior by using reverse engineering techniques. Reverse engineering aims to provide program descriptions on higher levels of abstractions. Since the UML [5] has been accepted as a standard, by the Object Management Group (OMG), it has become established as a principle means for modeling software systems. These models aid the program understanding process by allowing the software engineer to quickly gain a picture of the overall system.

Additionally, reverse engineering plays an important role in facilitating code reengineering and evolutionary development. This paper proposes the use of XML has become a common feature in development projects many developers have grown interested in integrating XML with the rest of the development.

Reverse engineering of class diagrams from existing software would be a significant advantage for development and maintenance. To analyze Java code, we use the parse method and to store parse result use the XML. In this system, we use XML Schema for document type. And we use parsing technique to analyze the code and then recover the model.

This paper is organized as follows: in section 2 related works with this system are discussed; section 3 and 4 describes the technologies involved in our approach are discussed; section 5 describes the implementation of the system; and finally we draw conclusion and highlight future directions for this paper.

2. Related Work

There are many research projects that are similar and related with our process. C.R Russell and R.G.Dewar presented XML encoded reverse engineering of java to uml exposing the relationship between an XML based representation of Java, namely JavaML and an XML based representation of UML, XMI [7].It stores the UML models with the XMI standard. The Precision Graphics Markup language (PGML) is used to save the graphical representation of models. Jorg Niere, Jorg P.Wadsack and Albert Zundorf presented recovering UML class diagram from Java code using pattern by describing the use of fuzzy pattern detection techniques for the recovery of UML collaboration diagrams from source code [3].

Nicholas Routledge and Linda Vird and Andrew Goodchild presented UML and XML Schema by defining a mapping between the UML class diagram and XML Schema using the traditional three level database design approach (i.e. Using conceptual, logical, and physical design levels) [4].There are many tools for program understanding. Rigi is a reverse engineering system that presents software subsystem structures using an interactive, multi-window graph editor and displays the source code through separate text editor windows. The Simple Hierarchical Multi-perspective (SHriMP) tool displays software architectural diagrams using nested graphs.

3. Background Theory

Reverse engineering is also support the analysis and understanding of data and processing in existing computerized system. The need for automated software maintenance tools and the role that reverse engineering techniques play in program understanding has been highlighted. Program understanding is the related term to reverse engineering. When a program is large and complex, we are very time consuming and difficult to understand the program structure. Moreover the software will be required to change in order to meet new requirements, improved technologies or some other need. Thus we build model so that we can better understand the system [1]. In addition, UML has been presented as a suitable modeling language for representing design abstractions. A tight mapping exists between UML and Object-oriented languages. Reverse engineering is the process of analyzing a subject system to:

(1) Identify the system components and their interrelationships and
(2) Create representation of the system in another form or at a higher level of abstraction. Such an abstract level could be e.g. a program description using UML diagrams.

For large legacy systems, the manual matching of such plans is difficult. One way of augmenting the program understanding process is through computer-aided reverse engineering. Reverse engineering is regularly applied to improve your own products, as well as to analyze a competitor's products or those of an adversary in a military or national-security situation. Although there are many forms of reverse engineering, the common goal is to extract information from existing software systems to better understand them. Its primary purposes are to provide an aid for comprehension or future redevelopment. There are six objectives for reverse engineering:

- to facilitate Reuse
- to provide Missing or Alternative Documentation
- to recover lost Information
- to Assist with Maintenance
- to migrate from one Hardware or Software Platform to Another
- to Bring the System under the Control of a CASE Environment.

There are many benefits by using reverse engineering: some of those are maintenance cost savings, quality improvement, competitive advantages and staff morale.

3.1. Software Maintenance

The ANSI definition of software maintenance is the modification of a software product after delivery to correct faults, to improve performance or other attributes, or to adapt the product to a changed environment, "according to ANSI/IEEE Std 729-1938.Usually, the system's maintenance was not its designers, so they must expend many resources to examine and learn about the system [2]. Reverse engineering is the part if the maintenance process that helps you understands the system so you can make appropriate changes [8].

To modify the system, existing program understanding is vital. A variety of reverse engineering tools provide to help program understanding. The overview picture of the system is important because it well gives the relationship between the elements and their components. Reverse engineering is not just about source code and data but it is also about complete system understanding.

4. UML and XML

UML is designed to allow models to be transformed into code and to allow code to be re-engineered back into models [2].The UML is a modeling language for object-oriented software systems with a strong emphasis on graphical representation. It is best suited for medium and large scale projects for two reasons:

(1) UML applies to Java, C++, Python, PHP, SQL, Web services, and just about any other development technology. It is easier to share designs across the team and

(2) UML diagrams can show as much as little information as necessary, so it is possible to prepare several models of increasing sophistication with the same tool.

XML is an efficient and effective way of storing information. Being a simple, we can use XML to store information, which can access XML on a wide variety of platforms. Information coded in XML is easy to read and understand, plus it can be processed easily by computers. The goal of XML is to be enabling the delivery of self describing data structures of arbitrary depth and complexity to application that requires such structures. Structurally, each XML document consists of a set of elements, the boundaries of which are

delimited by start-tags and end-tags. Each element has a type identified by name. Each attribute specification has a name and value. In addition, each element can have an arbitrary list of (nested) sub elements [9]. XML data structure and its syntax are defined in XML Schema (xsd). XML has numerous advantages including being easy to read, easy to parse, extensible, widely adopted and the data can be processed by other applications.

In UML, an attribute is a field attached to a class. In the java language, only one sensible mapping exists: the attribute becomes a class variable. In contrast, in XML the attributes may map either to a sub element or to a proper XML attribute. In Table 1 describe sample mapping from XML to UML use in our system.

Table 1. Sample Mapping from XML to UML

XML		UML	
<Class>		Class	
	<Modifier>		Visibility
	<Identifier>		Name
	<FieldDeclaration>		Attribute
	<FieldModifier>		Attribute Visibility
	<FieldType>		Attribute Type
	<Identifier>		Attribute Name
	<MethodDeclaration>		Operation
	<MethodModifier>		Operation Visibility
	<ReturnType>		Operation return type
	<Identifier>		Operation

5. Proposed System Design

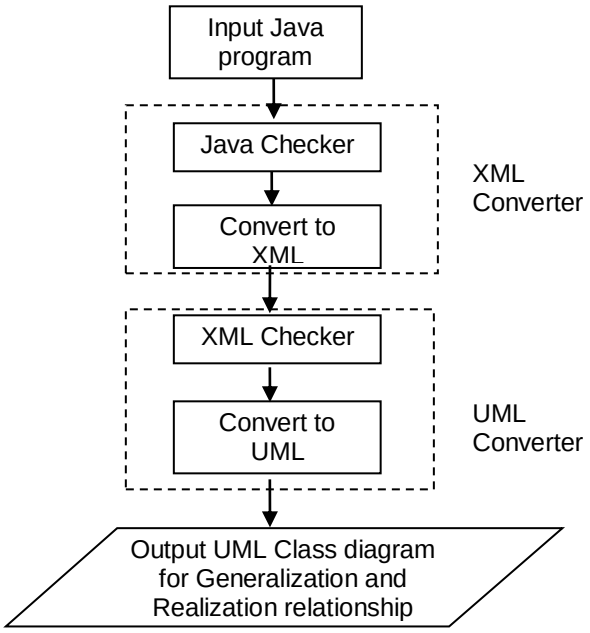

Figure 1. Propose System Design

System design is presented in Figure 1. Java provides the most robust set of APIs, parsers, processors, publishing frameworks, and tools for XML use of any programming languages. It is composed of two parts: XML converter and UML converter.

5.1. XML Converter

Firstly, XML converter contains two core classes. The first of these is code checker – to ensure the input java source code is error free code. The second class – Parser parse the checked source file using org.antlr.runtime package and use java JDOM API and xerces XML parser. The org.antlr package, which provides a framework for constructing compilers from grammatical descriptions containing actions in a variety of target languages. To create XML document from within Java applications use the JDOM API to construct the document and use the standard Java procedures to save the information in a file. Parse the java code and create an object array to store the parse information. And then, use the jdom API to create an XML document using the parsed information. In figure2 show the sample input code and converted XML file in Figure 3.

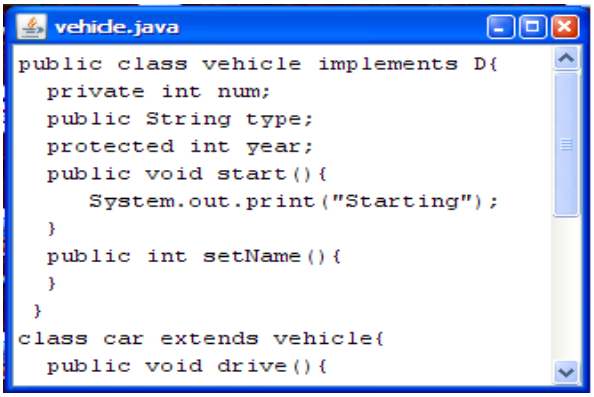

Figure 2. Input Java Program

5.2. UML Converter

In this part, use the output of the XML converter. Firstly it checks the validity and second, reading the XML files (parsing) and converts it to UML diagram. XML parser accesses (read) the XML documents from within the java code. An XML document is valid if its structure matches a defined schema. To ensure proper XML parser operation, we check XML document for validity before accessing them. After validation process, we parse the XML document using org.apache.xerces.parser.SAXParser and return a document object creating a tree in memory. And then access the parsed document using java API. If a return object is class, we extract the member of class and create a relevant UML diagram. Classify the content of document as which fields and operations contain in which class or interface and their relationships. We

classify the symbols of modifiers and return type in fields and methods.

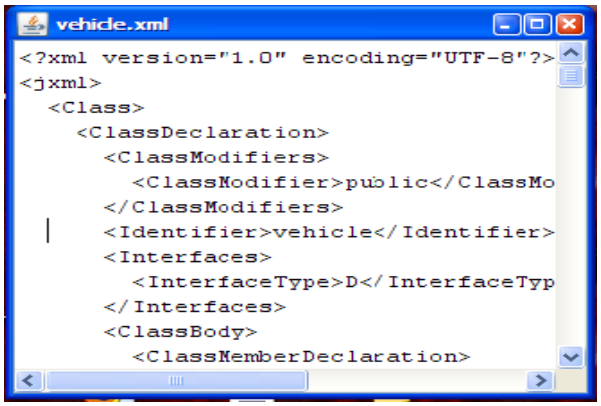

Figure 3. Converted XML File of Input Code

6. Class Diagram

UML defines nine distinct diagram types, divided into two groups. Four diagram types represent static parts of a system and five additional diagrams to view the dynamic parts of a system. Class diagram is a static kind showing the classes in a system and a variety of relationships between classes. Optionally, attributes and operation may be included. Class icons contain three compartments. The first compartment represents the name of class in java code. The second compartment represents the attributes of class and the third is operations. Optionally, a visibility, a type and a return type contain in attributes and operations. Simple class diagram of the output of UML converter is describe in Figure 4.

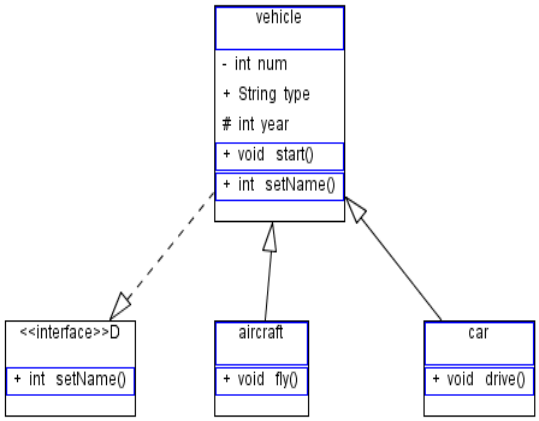

Figure 4. Output Class Diagram

6.1. Relationship

Relationship is a semantic connection among elements. UML defines four types of relationships: generalization, dependency, association and realization.

-Generalization relates one super class to one or more subclasses.

-An association between two classes implies the possibility of links between instances of the classes

-Dependency is a semantic relationship between two things in which a change to one thing may affect the semantics of the other thing

-Realization is a semantic relationship between classifiers, in which one classifier specifies a contract that another classifier guarantees to carry out [1].

In this system, the Java to XML parser checks the input word which match with the predefined extends keywords. The match word is mapped as Super elements in the XML file for generalization. The input word matches with the predefined implements keywords, the system map as Interface element in the XML file for realization. In XML data storage, the data is stored on a file which represents an XML Schema of the data. The data is stored in hierarchically. So, the system defines the generalization relationship between classes. If Class element's child is interface, we can define the realization relationship between classes.

7. Conclusion and Further Extension

To sum up, this reverse engineering system can be applied by system analyst and programmers. The class diagram is important in program's static structure description. This system can illustrate the visual class diagram. Thus, program design can be view and necessary modification can be made. This system enables the user to improve the design of existing code for software engineer. The user can easily get modifiable, understandable and robust form of the source code design. This system can only provide the program which is written in Java programming Language.

Significant of this system is the parse results are stored as XML file. XML is good for transmitting and formatting. It is also good for hierarchical data -- Data which has several levels of different sizes. So, we can easily define the member of class. This system can be made only generalization and realization relationship. Using this approach, more powerful reverse engineering tools can be developed. Developers can make Reverse Engineering on other Object-Oriented Programming Languages and to reversing the packages of Java program files with complex java program. Possibly the most significant extension to this system would be to development of Reverse Engineering tools. This paper proposes a method for designing program using the UML, which is widespread, and growing.

8. References

[1] G. Booch, J. Rumbaugh, and I. Jacobson, (2000): The Unified Modeling Language User Guide. Massachusetts: Addison-Wesley.

[2] E.J. Chikofsky and J.H. Cross II, “Reverse Engineering and Design Recovery: A Taxonomy”. IEEE Software, 7(1):13-17, Jan 1990.

[3] Jorg Niere, Jorg P.Wadsack and Albert Zundorf. “Recovering UML Diagram from Java Code using Patterns”, Department of Mathematics and Computer Science, University of Paderborn, WarbugerstraBe 100, 33098 Paderborn, Germany.

[4] Nicholas Routledge and Linda Bird and Andrew Goodchild, “UML and XML Schema”, School of Information Technology and Electrical Engineering, Distributed Systems Technology Center (DSTC), University of Queensland, QLD, 4072, AUSTRALIA

[5] OMG, editor. OMG Unified Modeling Language Specification, Version 1.3, June 1999. Object Management Group, http://www.omg.org, 1999

[6] R.S. Pressman. Software Engineering, A Practitioner’s Approach, McGraw Hill, 1997

[7] C.R. Russell and R.G.Dewar, “XML Encoded Reverse Engineering of Java to UML”, Department of Computer Science, School of Mathematical and Computer Sciences, Heriot-Watt University, Riccarton, Edinburgh, EH14 4AS, Scotland, UK.

[8] S.Ian Sommerville, “Software Engineering”, Sixth Edition. ISBN: 0-201-39815-X, Person Education Limited, 2001.

[9] V. Turau, “Making Legacy Data Accessible for XML Application”, web page, 2002, http:// www.informatikfhwiesbaden.del~tu/rau/veroeff.html.