International Journal of Combined Research & Development (IJCRD)
eISSN:2321-225X;pISSN:2321-2241 Volume: 4; Issue: 4; April -2015

Reverse Engineering of UML sequence diagram for the Abstraction of Requirements

Dr. R N Kulkarni[1], Singri Swathi [2], Afsana H [3], Soumya N M [4] , Heena Kousar [5]
[1] Professor , [2,3,4,5,] B.E. Final Year , [1,2,3,4,5] Dept of CSE,BITM Bellary, India

Abstract: Many modern software design methods have been developed to improve the reusability and maintainability of software and to reduce the time required for the maintenance and development operations, but many companies have old or legacy software systems, these companies spend a lot of money to maintain their old systems. These systems cannot be replaced by new systems so easily because they consist of lot business information which is accumulated over the years, and also there is no proper design information of the application is available.

In this paper, we are proposing an automatic tool for the abstraction of requirements from the input UML sequence diagram. The proposed tool abstracts the UML sequence diagram by parsing the UML sequence diagram and then the messages are abstracted and stored as scenarios which are then translated into requirements.

Keywords: Reverse engineering, UML sequence diagram, DOM parser, Abstraction, Scenarios, Requirements, XML

1. INTRODUCTION

Reverse engineering, also called as back engineering, is the process of extracting knowledge or design information from anything man-made and re-producing it or reproducing anything based on the extracted information. Reverse engineering is a process of examination only: the software system under consideration is not modified. It can also be seen as "going backwards through the development cycle".

In reverse engineering the process is often tedious but necessary in order to study the specific technology or device. In system programming, reverse engineering is often done because the documentation of the particular system has never been written or the person who developed the system is no longer working in the company. We use this concept to introduce an automatic tool for retrieval of requirements of a system from the UML sequence diagram.

The purpose of proposing this tool is to be able to recover the system requirements of any system due to the cause that the system does not have the necessary documents. Documenting the process involved in developing the system is important. In many organizations, twenty percent of system development cost goes to documenting the system. In software development life cycle (SDLC), documenting the system requirements analysis ends with a system requirements document (SRD). SRD is important in order to develop a system. It shows the system specification before a developer would be able to develop the system. Once the system demonstrates faults after design phase, the SRD can be used as a reference for identifying errors of the system requirements. This is a difficult process considering the large sequence diagrams. Therefore by having a tool that would be able to retrieve the system requirements back from the UML sequence diagrams would be an added advantage to the software developers of any system application.

2. RELATED WORK

Ashalatha Nayak et al., [1] have proposed an approach of synthesizing test data from the information embedded in model elements such as class diagrams, sequence diagrams and OCL constraints. Scenarios derived from the sequence diagrams describe the functionality of a system under development in terms of its behavioral descriptions. In their context of system testing, scenarios representing an abstract level of test cases need to be augmented with test data. A SCG is a directed graph which is obtained by integrating the necessary information from a class diagram, OCL constraints and a sequence diagram. The idea was to express the underlying control flow information involved in a sequence diagram as a directed graph. The sequence diagram based coverage criteria were used to generate a set of scenarios which are to be covered during testing. Here, SCG graph is used, which requires more processing.

Philip Samuel et al., [6] have implemented a method for generating test cases automatically from UML sequence diagrams in a prototype tool named UTG. Here, UTG stands for UML behavioral Test case Generator. UTG has been implemented using Java and can easily integrate with any UML CASE tools like MagicDraw UML [4] that supports XML (Extensible Markup Language) format. Since UTG takes UML models in XML format as input, UTG is independent of any specific CASE tool. They have used the tool with several UML designs. In our implementation, we make use of XML format to represent UML sequence diagram as it is easier to understand and it also represents all aspects of sequence diagram.

Manar H. Alalfi et at.,[3] presented an approach and tool named PHP2XMI to automatically instrument dynamic web applications using source transformation technology, and to reverse engineer a UML sequence diagram from the execution traces generated by the resulting instrumentation. PHP2XMI automatically generates XMI sequence diagram files which can be visualized directly in any UML toolset.

The approaches on generating test cases [6] or test data [1] from sequence diagrams and on extracting sequence diagrams from execution traces [3] were proposed. A novel methodology that uses the concept of slicing UML diagrams using model based slicing technique has been proposed [8]. But there is not such approach proposed for abstracting requirements from sequence diagram. Hence our motivation is to build a tool which abstracts requirements from the given sequence diagram.

3. TERMINOLOGIES

3.1 Reverse Engineering:

The reverse engineering is the process of analyzing the subject system with two goals:

-To identify the systems components and their interrelationships

- To create representations of the system in another form at a higher abstraction level.

3.2 UML:

The Unified Modeling Language (UML) is a general-purpose modeling language in the field of software engineering, which is designed to provide a standard way to visualize the design of a system.

3.3 Sequence Diagram:

A Sequence diagram is an interaction diagram that shows how processes operate with one another and what is their order.

3.4 Scenario:

-A scenario is a description of a person's interaction with a system.

-Scenarios help focus design efforts on the user's requirements, which are distinct from technical or business requirements.

-Scenarios may be related to 'use cases', which describe interactions at a technical level

3.5 Requirements:

These are prerequisites for the system to work as intended.

3.6 Restructuring:

Restructuring is the transformation from one representation form to another at the same abstraction level. The transformation preserves the external behavior of the system. Restructuring here is used in implementation stage to transform code from an unstructured form to a structured form [7].

4. PROPOSED METHODOLOGY

We are proposing an automated tool to reverse engineer the UML sequence diagrams for abstraction of requirements. The following steps to be followed to abstract requirements from UML sequence diagram

Algorithm
Input: UML Sequence diagram
Output: Sequence of requirements

1. [Input / Draw the UML Diagram]
   1.1 Draw UML sequence diagram using any CASE tools like Visual Paradigm or MagicDraw or Rational Rose

2. [Restructuring]
   2.1 Scan the sequence diagram from left to right and top to bottom
   If a message is not having arrow mark
   Then
   Insert proper arrow
   Else
   If Message name is not present
   Then
   Insert proper message name
   End if
   End if
   2.2 Align each message in a time order

3. [Store the Entire UML sequence diagram]
   3.1 Save the UML sequence diagram in Visual Paradigm with .vpp extension.

4. [Export the Sequence diagram into its equivalent XML file]
   4.1 Visual paradigm for UML 12.0 version provides the in-built functionality to export the diagrams into XML format.
5. [Parse the XML file]
   5.1 Java API DOM is used to parse the XML code file generated in step 4.
   5.2 DOM parser uses the function DocumentBuilderFactory() to create the instance of the class to parse the file.
   5.3 DOM parser will generate a txt file having information regarding object name and its identifier. This file also contains the information related to all the messages and the objects among which the message is floating.
   5.4 All the information generated by parser will be stored in separate .txt file.
6. [Extract scenarios]
   6.1 Read .txt file created in step 5 and extract the messages, objects identifier, message type, message to & from information in sequential order of messages.
7. [Convert Scenario to requirements]
   7.1 Convert the extracted scenarios into requirements by constructing English sentences.

5. CASE STUDY

Consider an example sequence diagram of System interacting with user and database (see figure 5.1) to generate requirements.

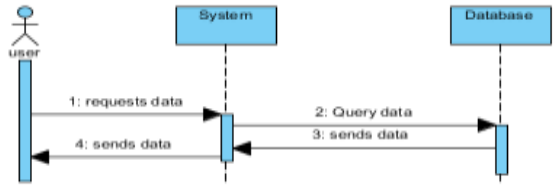

Figure 5.1 UML Sequence Diagram

The sequence of requirements generated as shown in figure 5.2.

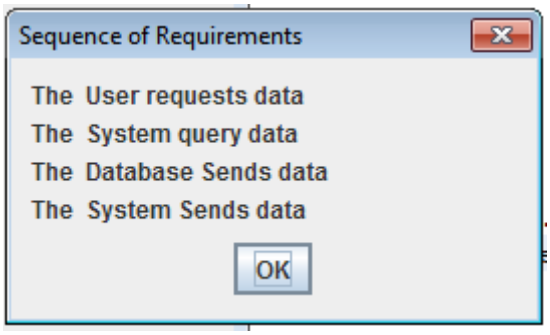

Figure 5.2 Sequence of requirements

6. CONCLUSION

We have proposed a new method for the abstraction of requirements from UML sequence diagram. And these abstractions are carried out through a sequence of steps like restructuring, parsing, abstraction of messages and storing them in the form of scenarios and finally these scenarios into the requirements. So we have tested the tool for its correctness and completeness.

REFERENCES

[1] Ashalatha Nayak, Debasis Samanta: "Automatic Test Data Synthesis using UML Sequence Diagrams", in *Journal of Object Technology*, vol. 09, no. 2, March-April 2010, pp. 75-04, http://www.jot.fm/issues/issue 2010 03/article2/

[2] Heumann J (2001) "Generating Test Cases from Use cases, Rational Software, IBM"

[3] Manar H. Alalfi James R. Cordy Thomas R. Dean "Automated Reverse Engineering of UML Sequence Diagrams for Dynamic Web Applications", School of Computing, Queen's University, Kingston, Canada falalfi, cordy, deang@cs.queensu.ca

[4] Mohd Hafeez Osman and Michel R.V Chaudron "Correctness and Completeness of CASE Tools in Reverse Engineering Source code into UML Model", LIACS, Leiden University Niels Bohrweg 1, 2333 CA Leiden, the Netherlands {hosman,chaudron}@liacs.nl

[5] Musker D (1998) Reverse Engineering IBC Conference on protecting and exploiting intellectual property in electronics

[6] Philip Samuel, Rajib Mall "A Novel Test Case Design Technique Using Dynamic Slicing of UML Sequence Diagrams", in e-Informatica Software Engineering Journal, Volume 2, Issue 1, 2008

[7] Dr. Shivanand M. Handigund & Rajkumar N. Kulkarni, "An Ameliorated Methodology for the design of Object Structures from legacy „C" Program" International Journal of Computer Applications (0975- 8887),Volume 1, No. 13, March 2010, Page No. 61-66.

[8] Singh et al., "Technique for Extracting Subpart from UML Sequence Diagram "International Journal of Advanced Research in Computer Science and Software Engineering 3(6), June - 2013, pp. 593-596

[9] Sommerville I (2007) Software Engineering, 8th edition, Addison Wesley, England