International Journal of Scientific & Engineering Research, Volume 3, Issue 11, November-2012
ISSN 2229-5518

Development and Analysis of RE Tool for Recovering Design Artifacts during Software Reverse Engineering

Parul Dongre, Arvind Upadhyay

Abstract—Reverse engineering provides a direct attack to solve comprehension problem of large software system. This research paper elaborates development of tool for reverse engineering of java code & recovering the concept to discovering the design artifacts of a software system from its source code and related documentation. Here we used REAM (reverse engineering abstraction methodology) for tool development and evolve algorithm which reverse engineer the java code to generate class diagram, use case diagram, activity diagram. There are already some tools available in market, but tools are having very limited reverse engineering capabilities. Like, they do not clearly specify relationship between classes, because all the relationships are not properly extracted. Here we have developed a tool named 'RE Tool' which tends to provides all reverse engineering capabilities. As well as after development, analysis is done by analysing two different commercial tools ArgoUML and Altova Umodel@2012 with RE Tool considering case studies.

Index Terms— Reverse engineering; level of abstraction; legacy application; REAM; Class diagram ; ArgoUML ;Altova Umodel@2012.

1 INTRODUCTION

The term reverse engineering as applied to software means different things to different people, prompting Chikofsky and Cross to write a paper researching the various uses and defining taxonomy. From their paper, they state, "Reverse engineering is the process of analyzing a subject system to create representations of the system at a higher level of abstraction."[1].

A common problem experienced by the software engineering community traditionally has been that of understanding legacy code. Reverse Engineering is a methodology that greatly reduces the time, effort and complexity involved in solving the program comprehension problem. Reverse Engineering is best defined by Chikofsky and Cross as "the process of analyzing a subject system [1]-

- To identify the system's components and their interrelationships and
- To create representations of the system in another form or at a higher level of abstraction.

During the reverse engineering process, the source code is not altered, although additional information about it is generated.

.

- *Ms.Parul Dongre is currently pursuing masters degree program in computer science & engineering in IES, IPS academy, Indore, India. E-mail: parul.dongre@gmail.com*
- *MR.Arvind Upadhyay is Assistant professor in department of computer science & engineering in IES, IPS academy, Indore, India. E-mail: upadhyayarvind10@gmail.com*

2 METHODOLOGY

A key success to reverse engineer the code and find design artifacts is well defined and precise methodology. Here we are using reverse engineering abstraction methodology (REAM),[9] it comprises five models which are high level model, functional model, architectural model, source code model and mapping model. High level gives an abstract understanding of the system at a higher level to recover the artifacts. Functional model represents the functional elements and relationships among the system artifacts. Source code model is extracted from the source code using the high-level model (and functional model) to develop an understanding and model it at an abstract level. The architecture model is developed with the understanding gained out of developing the high level, functional and source code models and by understanding the dependencies between the various artifacts.

As figure no. 1 describes the five models used in REAM methodology [9]. The reverse engineering tool communicates with all these five models to generate design artifacts. Which facilitate further user to understand legacy codes, which are having great complexity.

The user defines a mapping model between the entities in the source code model (i.e., functions, classes) and the entities in the high-level (i.e., concepts), functional (i.e., relation and logic among programs or concepts) and architectural (i.e., modules or components) models.

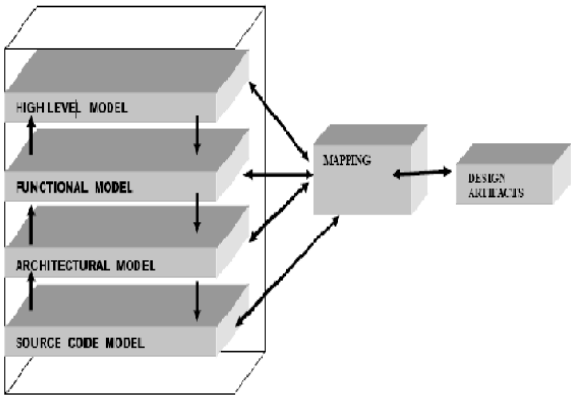

Fig. 1. REAM Model

3 PROPOSED SOLUTION

Reverse engineering is the process of analyzing a subject system to create representations of the system at a higher level of abstraction [1]. It can also be seen as going backwards through the development cycle.

Reverse engineering can be viewed as the process of analyzing a system to identify the system's components and their interrelationships and to create the representations of the system in another form or a higher level of abstraction and to create the physical representation of that system [1].

Many research groups have focused their efforts on the development of tools and techniques for program understanding. Software reverse engineering, or program comprehension, is the difficult task of recovering design and other information from a software system. It is difficult to perform because there are intrinsic difficulties in performing the mapping between the language of high level design requirements and the details of low level implementation [10]. The major research issues involve the need for formalisms to represent program behaviour and visualize program execution. Reverse engineering has many supporting aspects [2]. It may focus on features such as control flows, global variables, data structures, and resource exchanges. At a higher semantic level, it may focus on behavioural features such as memory usage, uninitialized variables, value ranges, and algorithmic plans. At an even higher level of abstraction, it may focus on business rules, policies, and responsibilities. Each of these points of investigation must be addressed differently.

4 DEVELOPMENT METHODS

We will actually design a parser that will implement the concept of reverse engineering and accordingly reversely engineer the input java code & generate the respective design artifacts like class, sequence, activity and use case diagram. In order to implement the concept of reverse engineering we firstly allow the user to import the desired input java code into the interface (Input Screen where he can import the java code from the stored path) and then step by step generate the class diagram first after that we proceed towards the sequence, activity and use case diagram.

It is clear from the System Interfaces point that we are allowing the user an interface from which he can able to generate the required design artifacts after reverse engineering the java code. At first user will interact with a screen. Here he defines the source path of the input java code file. After defining the source path of the java code and importing it the actual process starts.

The base guideline for the research is Reverse Engineering concept. The entire algorithm in most general form of understanding is defined in this point. It will actually describe that how the input java file isprocessed

Class Diagram Generator: The first solution came in the form of "Class Diagram". Here there is another interface by which the user can define the settings.

parameters of the class diagram like in terms of hierarchical degree of separation, in terms of functions and attributes of each sub classes (child classes) also the root (parent class) etc. There are some more properties are there which the user can set according to his need.

Sequence Diagram Generator: After successfully generating the class diagram the next will be to generate the desired sequence diagram. Here we used parsing technique to generate the sequence diagram.

Activity Diagram Generator: The next step is for the generation of activity diagram. For this we need to extract the loops and operators used in various functions of the classes. Then we can find the various activities.

Use Case Diagram Generator: After activity diagram, the next step is to generate the use case diagram of the java code. This is a very crucial step. Use Case diagram successfully depicts all the functionalities as well as the interaction between various actors.

5 ALGORITHAMS TO GENERATE CLASS DIAGRAMS, USE CASE AND SEQUENCE DIAGRMAS

5.1 To generate UML design artifact: Class Diagram

Input: Java Source Files
Output: Class Diagram

1. Repeat while all files are analyze
2. javaClass ← class files related to input so //list all class files in the input code
3. relation ← temporary saves the relationship of class
// get the class type (like extended…) map the relation
4. nameFunction ← get names of functions in classes
// retrieve member function in class map the function
5. nameVar ← get variables used in function
// get the members
6. return type ← get return type of variable
// get the return type of function map the type and variables
7. Call DrawVariable ()
// graphics function to draw the image

5.2 To generate UML design artifact: Activity Diagram

Input: Java Source Files
Output: Activity Diagram

1. Parse java file
   // add match function to parse file
2. Find Loops
   // find all for, while, do…while structure
3. Find Decision Operators
   // find all if, Else, switch and case statements
4. Find Loop Contents
   // find code block
5. Find Decisional Contents
   // find code block
6. Call Draw Function
   // to draw java files.

5.3 To generate UML design artifact: Use Case Diagram

Input: Java Source Files, No. of Actors, Actors relation (Chooses by user itself)
Output: Use Case Diagram

1.JavaClass ← class files related to input source
//list all class files in the input code
2. Relation ← temp save relationship of class
// get the class type (like extended…)
map the relation
3. Call draw function
//draw a basic diagram
4. Add actors
// user defined actors
5. Call draw function
// call draw function to add actors
6. Add relationship
// user prepare a relation between user and classes
7. Call draw function
// draw the relations

6 RESULT FROM THE TOOL

Above is a description of how the system will interact with its users. It actually defines how the user will interact with the application which uses the concept of reverse engineering and extract the design artifacts from the input java code & helps us to understand the working the functionalities of the code in a more easy and understandable manner.

After development of tool (RE Tool) we are analyzing our tool with Altova Umodel@2012[4, 5] and ArgoUML [6, 7]. Here for this we consider two different java codes which are airthmatic24 game [3] and college database application. Airthmatic24 game is developed in Java by Huahai Yang. It is a simulation of popular traditional card game. The second code which is under consideration is small college database application. There are also some research have been done on two most successful industrial-strength CASE-tools (Together and Rose) in reverse engineering the static structure of software systems and comparison done between them. To estimate the quality of the diagrams more objectively, they also compared with a "correct class diagram", which could be a design model or a model manually constructed by the experts of the subject system [11]. Same approach is followed here during this research work.

Below there are snapshots by considering two java programmes run on respectively ArgoUML and Altova Umodel@2012and RE Tool (which is developed during this research work).

A. ArgoUML

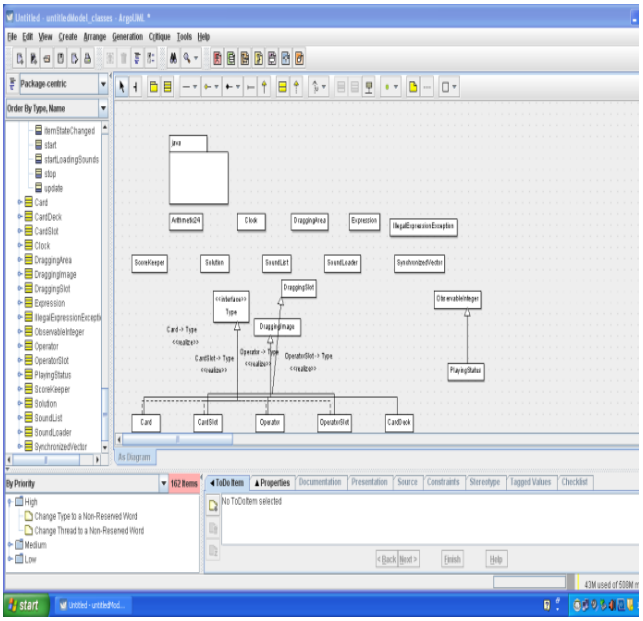

Fig 2. Arithmetic 24 game by ArgoUML

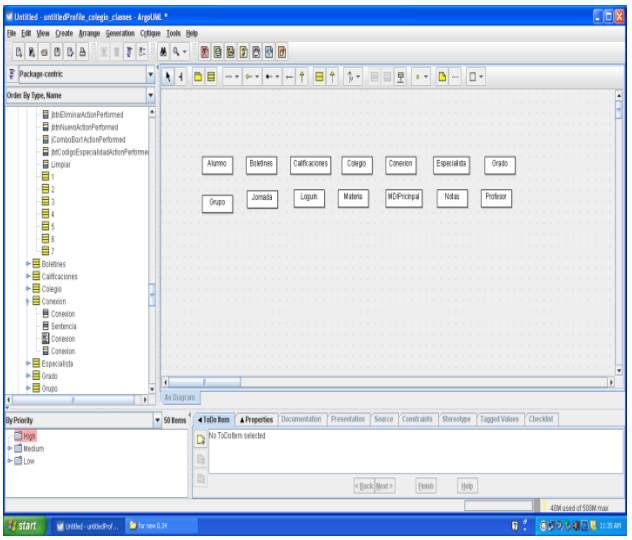

Fig 3. College DB application by ArgoUML

B. Altova Umodel@2012

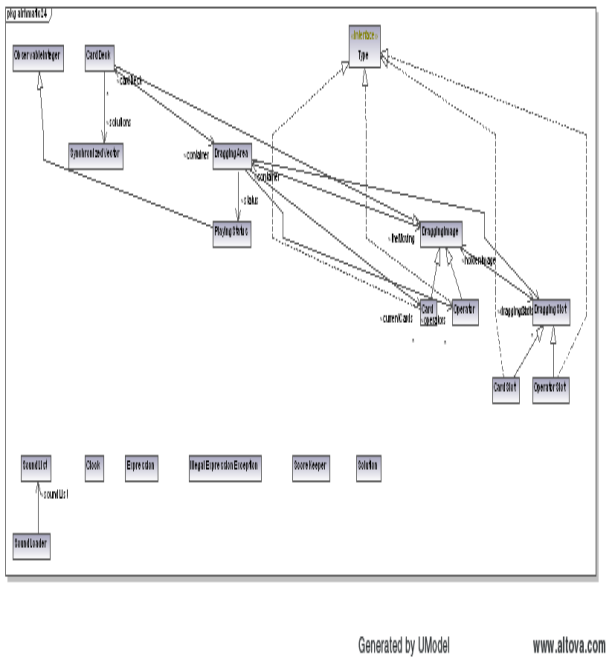

Fig 4. Arithmetic 24 game by Altova Umodel@2012

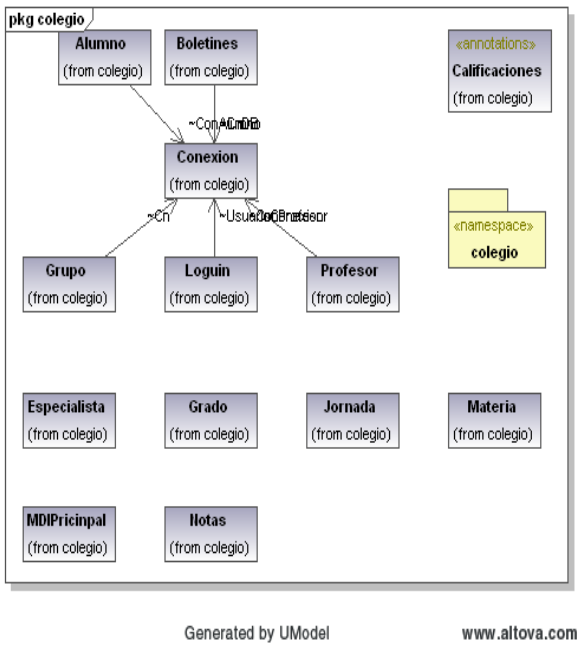

Fig 5. College DB application by Altova Umodel@2012

C. RE tool

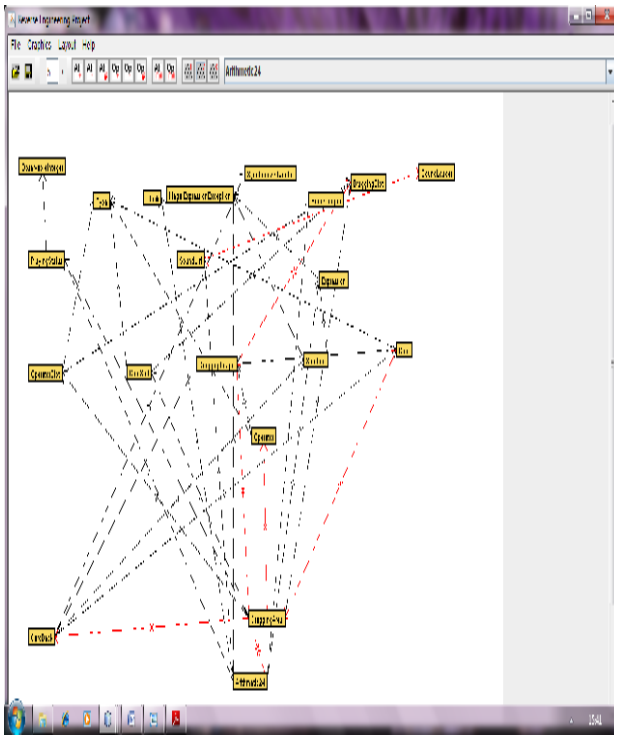

Fig 6. Arithmetic 24 game by RE Tool

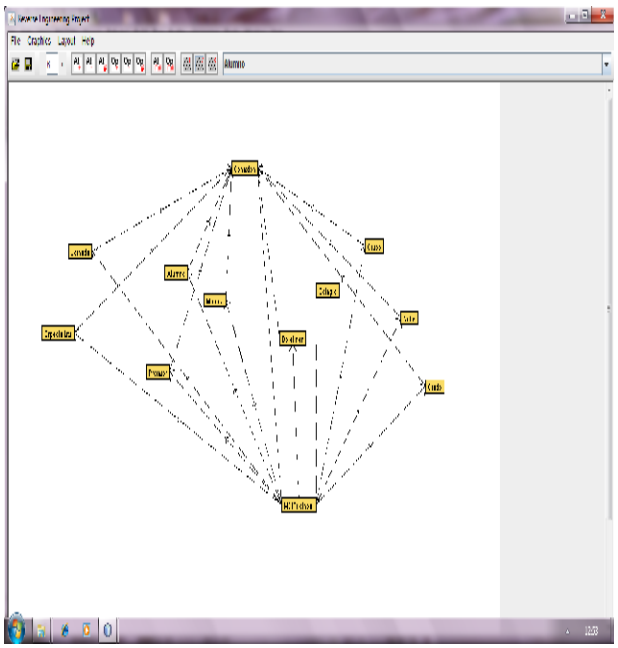

Fig7. College DB application by RE Tool

For ArgoUML by importing java code of airthmatic24 game and selecting the 'class diagram generation' option screen (Fig .2) is generated. It generates total 19 classes from java code and zero number of associations because it not recognizes association in code. By importing java code of college database application and selecting the 'class diagram generation' option screen (Fig. 3) is generated. It generates total 14 classes from java code and zero number of associations.

For Altova Umodel@2012 after importing java code for airthmatic24 game by clicking tree tab. And after Expanding the Class Diagrams package, all class diagram contained in the project are displayed (Fig. 4). By double click the main diagram icon the Class diagram appears as a tab in the working area. After class diagram generation diagram, it identifies that it recover total 19 classes and 11 associations. By selecting java code of college database application, it generates total 13 classes from java code (Fig. 5) and 5 associations.

For airthmatic24 game RE Tool (Fig. 6) recovers total 19 classes and 22 associations. For college database application this tool recovers 14 classes and 21 associations (Fig. 7).

For analysis here we consider two parameter which are number of classes and number of association. Numbers of classes (NOC) are the general measure for the overall size of a software module. Therefore, high NOC values may indicate a more detailed representation [8]. Second parameter is number of association (NOA) which is a metric measure of interconnectedness in a module [8]. In reverse engineering it is important to understand how classes are connected.

TABLE 1

PARAMETERS FOUND BY VARIOUS TOOLS

Project Name	Tool Name	No. of classes	No. of Association
Arithmetics 24	Argo UML	19	0
	RE Tool	19	22
	Altova Umodel	19	11
College Database application	Argo UML	14	0
	RE Tool	14	21
	Altova Umodel	13	5

7 CONCLUSION

Understanding object-oriented programming can be a difficult task no matter what language are used. For Program understanding we can use various kinds of tools which support reverse engineering to recover lost information, for improper documentation, to provide alternate view, to extract reusable components and to detect side effects. The purpose of this research paper is to develop a tool which gives a facility to recover design artifacts from java source code. Which is helpful when there is general lack of understanding of system exist, if document is unavailable, if staff member left the job or any new people in the staff. This research paper elaborate development of reverse engineering tool which we gave name 'RE tool' for java code which further recover the design artifacts of a software system from its source code and related documentation. Here we also analyzed this tool with two commercial used software tools like ArgoUML and Altova Umodel@2012 by considering three different case studies.

REFERENCES

[1] Chikosfy E.j. & cross J.H., "Reverse engineering and design recovery: A taxonomy".IEEE Software, january 1990 ,7(1):13-17

[2] Hausi Muller, Kenny Wong, Scott Tilley, "Understanding Software Systems Using Reverse Engineering Technology, Colloquium on Object Orientation in Databases and Reverse Engineering" The 62nd Congress of "L' Association Canadiene Francaise pour l' Avancement des Sciences (AFCAS)",May 16-17

[3] http://javaboutique.internet.com/arith24/ :- „Arithmetic 24 Game, which is developed in Java by Huahai Yang. It is a simulation of popular traditional card game.

[4] Altova Umodel information from http://v2006.sw.altova.com/UModel.pdf

[5] Altova Umodel tutorial from http://manual.altova.com/umodel/umodelbasic/index.html?umtutorial.htm

[6] Argo UML manual from http://argouml.tigris.org/files/documents/4/8727/argomanual.pdf

[7] ArgoUML tutorial from http://staff.ustc.edu.cn/~zhuang/cpp/tools/ArgoUML%20Tutorial.pdf

[8] Shivani Budhkar, Dr. Arpita Gopal "Reverse Engineering Java Code to Class Diagram: An Experience Report" International Journal of Computer Applications 29(6):36-43, September 2011. Published by Foundation of Computer Science, New York, USA.

[9] Nadim Asif "Reverse Engineering Methodology to Recover the Design Artifacts: A Case Study",In Software Engineering Research and Practice 2003 pg no.932-938

[10] Michael L. Nelson "A Survey of Reverse Engineering and Program Comprehension" ODU CS 551 - Software Engineering Survey, April 19, 1996

[11] Ralf Kollmann, Petri Selonen, Eleni Stroulia, Tarja Syst and Albert Zundorf "A Study on the Current State of the Art in Tool-Supported UML-Based Static Reverse Engineering", Proceedings of the Ninth Working Conference on Reverse Engineering (WCRE.02) 2002 IEEE.