*World of Computer Science and Information Technology Journal (WCSIT)*
*ISSN: 2221-0741*
*Vol. 3, No. 1, 4-7, 2013*

# Framework Environment for Understanding Object Oriented Techniques

Hamed J. Al-Fawareh
Faculty of Science and Information Technology
Zarqa University
Zarqa, Jordan

Abstract — Source code analysis does not provide enough information for the program understanding activities. Understanding object oriented software systems during software maintenance are difficult and take a considerable amount of time and cost. In software maintenance perspective reverse engineering process extracts information to provide visibility of the object oriented components and relations in the software that are essential for maintainers. Various tools have been developed to help maintainers undergo this process. One such tool is reverse engineering object Oriented program tool. In this paper, we provide a framework environment for object oriented programs. The framework focuses on the technical side of maintaining object-oriented program. This paper will discuss the development of the tool and highlight a scenario to show the viability of the tool by providing querying and browsing facilities.

Keywords - Software Maintenance, Reverse Engineering, Object Oriented.

## I. INTRODUCTION

The advancement of technology has caused a rapid change in computer software and hardware. Many software applications demand software developers to build large software systems that contain thousands of components, which may spread in different and large modules. Consequently, it may be difficult to determine changes into these modules and sometimes the changes may introduce unexpected interactions between diverse parts of the software systems.

Previous work in software maintenance focus on maintenance of programs written in traditional programming language such as C and Fortran [1, 2, 3, 4, 5,].Object oriented feature like data encapsulation that effectively address many exist maintenance problems. But new features like polymorphism, dynamic binding, message passing and overloading pose new problems for maintainers. Early research has shown that many issues in maintenance of object oriented are yet to be unraveled.

Many maintenance activities involve analyzing the relationships or dependencies that exist among various components of a program. Program dependence analysis is used to identify various types of relationships among the components.

## II. FRAMEWORK ENVIRONMENT

In this section, we presents a framework for object oriented programs, which will become of an environment for maintaining object oriented system. This framework focuses on the technical side of maintaining object oriented program, and thus may be viewed as a model of a larger maintenance environment. This larger maintenance environment may be embedded in a software-engineering model, which may be embedded in the corporate model.

The process in the framework is depicted in Figure1. A source code modification request arrived. It may be adaptive, perfective, corrective or preventive. To make the change or enhancement in the software system the maintainer needs to understand the source code and the statement that change will occur on it. Also, the maintainers need to know all the relationships and dependencies between the software components.

The proposed framework environment design for understand the software system by giving all the relationships and dependencies that occurs between the software components. These information will feature as relations which will used by the maintainers to understand the certain part of the software system, which is the most important part to modify and change. In order to reduce time, cost and maintainers effort, framework environment gives the relationships and

dependencies captured, so the maintainers will know all the parts effected by the change and enhance part, so this process will reduce time, cost and the maintainers effort.

The proposed framework assists the maintainers in understanding system constituents and their relationships as well as in associating between object oriented components in object oriented software systems.

The process in the proposed framework as shown in Figure 1, When link point system reads the source code, modifying the software, which requires two steps. First, understanding the source code, which may require documentation, code reading and execution. Secondly, when the program is modified. The framework process represents the relationships between object oriented program in two forms textual and dependence graph.

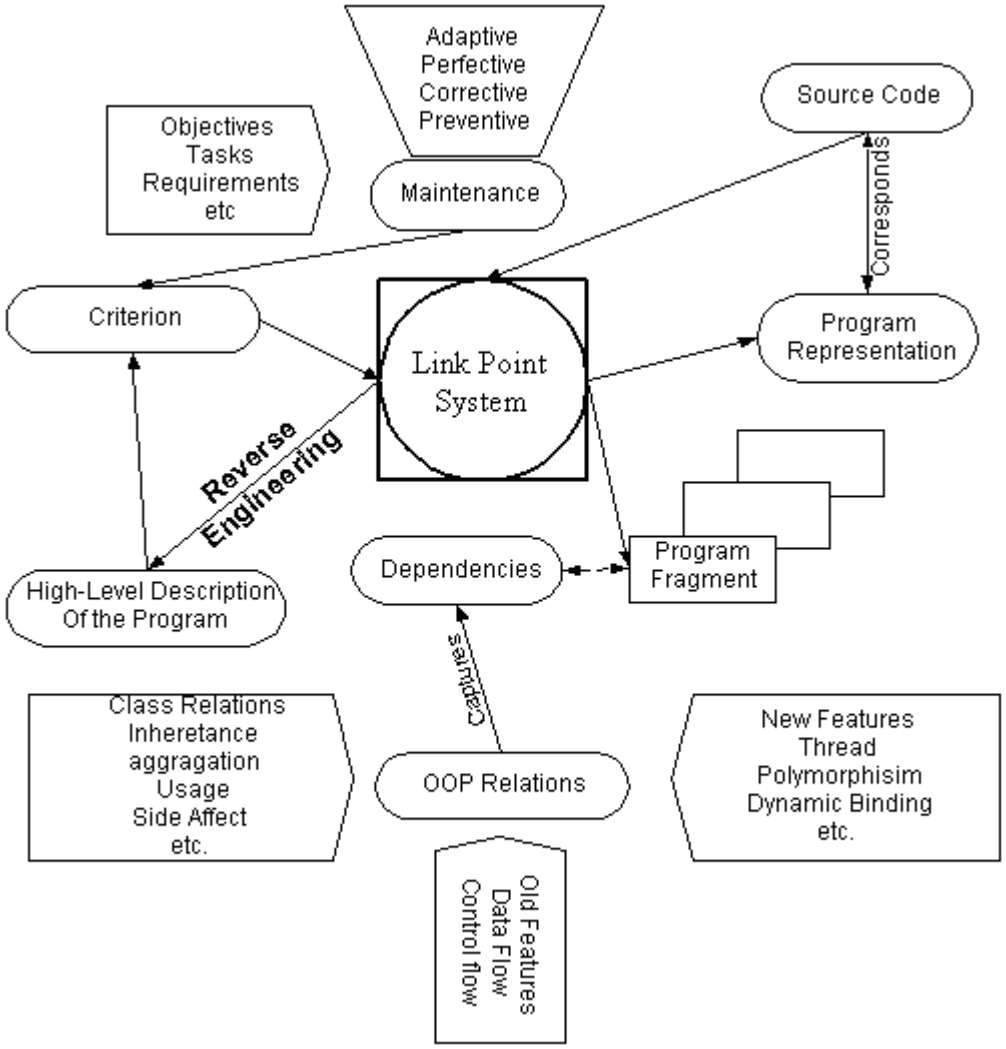


Figure 1: System Framework

## III. REVERSE ENGINEERING APPROACH

Managing and maintaining of software systems require the availability of information knowledge and abstraction level related to the intended objectives and their associated operations. From the software maintenance perspective, reverse engineering is the process of extracting information concerning software product design. Information complemented is essential for a developer and maintainer, when a system processes large and huge modules. This information aids humans to understand by representing a complete visualization of the intended objectives[6-7].

Moreover, reverse engineering gives facilities that help to control many managerial problems. These problems occur when the developer builds a large software system, which contains thousand of statements and thousands of functions[8]. For example, a database contains thousands of modules and many thousands of dependency relationship documents and histories. It is beyond the ability of the maintainers to manually search this database to find a specific module or variable, its dependency relationships, and all the

activities in these products. Maintainers usually spend more time reading and modifying huge and complex software systems than building them.

Thus, the tasks involved in reverse engineering are analysis of a subject system process which is identifying systems components and their interrelationships, in addition to creating representations of the system at higher level of abstraction [3]. The maintainer spends a lot of time to understand all the activities of the problem and to correct it. These involve scanning the source code, reading the documents, and understanding how to make necessary changes. The aim of reverse engineering is to remove ambiguity in the software and understand the software system in different programming languages with the aspect of facilities, enhancements, redesign, and correctness [9-11].

## IV. Approach Taken

The link point system assists in visualizing system constituents and their relationships, and in binding between object oriented program components. In addition, it provides information with the objective to help users to understand various relationships in the object oriented program.

The proposed link point system contains three modules. These models are SCANNER MODULE reads the Java source character by character, and constructs the symbol table. The scanner builds up the tokens of the program. The PARSER MODULE accepts the source code as an input, and produces a parse tree. The EXTRACTOR produces system structures. The system structures represent a component of the software system as n-tuple, the Cartesian product of domains d1,. . . , dn which is denoted by $d1 \times d2 \times ... \times dn$ for $n>1$, and the sets of all tuples (X1,...,Xn) such that, for any i=1,...,n, ($Xi \in$ Di ). Furthermore, the EXTRACTOR module automatically extracts the information from the software system and transfers these information to the data base.

The system structure given a data structure is defined as a digraph G of an ordered pair G=(C, R) where C denotes a program components and R denotes a component relations, which belong to Cartesian product $D \times D$, where C and R are finite sets. The data structures that are automatically transfer into the parse tree. The user interfaceaccepts questions posed by users and browses the data-based module depending on the structure of the questions posed. The system structure contain relations between all the components of thesoftware system. These relations can be extracted from the source code of the Java language directly.

### A. Case Study

A lot of case studies where performed in order to demonstrate the viability of the proposed system. The case studies with the Java programming language. The names of the classes and their components are the objects of maintainer. The maintainer may also need to know where to declare the objects, and may also need to know which classes are used in a specific location.

It is beyond human ability to search manually a database, which contains thousand of dependency relationships, and find specific modules, their dependencies, and their connectivity with other activities in the life cycle of a product. Furthermore object-oriented technique features a new dependencies occurs between a program components, that is class, message, methods and variables.

Linked point system help maintainers to understand a java information about the object oriented system written in Java language. These information include program components that is classes, method, variables and message. Furthermore linked point take into account the dependencies occurs in object oriented techniques and gives enough information about inheritance, polymorphism, dynamic binding and encapsulations. The linked point shows a class relationships diagram as a designed representation, it also represent the class relationships as implemented representation.

The linked point provide in this paper gives a facility for a maintainer to ask about the object oriented system components not only the declarations or the statements that a specific variables occurs but also the maintainers can ask about the kind of and the statements condition.

The system also, provides report facilities. To illustrate how the user interacts with the report system. The report can be issued as file or class report . Where the file report contains the total number of files that have been scanned by linked point application and total classes, attributes and methods. Class report represent classes methods, and attributes, which represent the main objectives of the research that is to discover the relationship between “as implemented” and “as designed” architectures.

For example maintainers may ask which object affects a specific variable with specific conditions.

Which function called by the module Q?
Another query is presented as follows:
Given the life cycle of the variable x?

This query can be respond to in the logic form reverse engineering system by using the target “variables “ in the main menu, then by typing the variable name and the scope.

## V. CONCLUSION

Reverse engineering is a process of extracting information from a software system. This information reduces the software system complexity, which responsible for the incorrectness of the software system and makes it different to understand object oriented system and detect bugs. According to the proposed system interface connections between Java source

code language classes, methods and variables to help us in the above mentioned task. The linked point system extract the information from the Java source code and produce internal structure, which will in twin facilitate the job of the object oriented software system of the automatic aids.

Linked point system provides technique to help users, in different way to understand an object oriented system and detect bugs by getting all the information about the classes, methods and variables, starting from the tie when users declare it in the program until the end of the class, method and variable life cycle. The proposed system can be extended and implemented in other object oriented languages. Also, the system can be implemented in metric, measurements, retrieving, and abstraction.

## REFERENCES


[1] Goyal, A. , Sharma, P. Goyal, S.B. and Singhal, N. (2012) "Analyzing Object Models with Theory of Innovative Solution" Second International Conference on Advanced Computing & Communication Technologies (ACCT), 2012 pp 46 - 50

[2] Liang Bao, Chao Yin; Weigang He; Jun Ge and Ping Chen (2010) "Extracting reusable services from legacy object-oriented systems" IEEE International Conference on Software Maintenance (ICSM), 2010 pp.1 - 5

[3] Gerardo Canfora, AnielloCimitile, A Logic-Based Approach to reverse Engineering Tools Production, IEEE Transactions on Software Engineering, Vol. 18. No. 12, 1992.

[4] Huan Li; Beibei Huang; Jinhu Lu, (2008) "Dynamical evolution analysis of the object-oriented software systems ", IEEE Congress on Digital Object Identifier, pp. 3030 – 3035

[5] Khaled khan Md., and TorbjornSkramstad, , (1995), "Data Structures for extracting and preserving low-level program information", Malaysian Journal of Computer Science, Vol. 8, No. 2, pp. 66-79.

[6] ChikofskyEllot J. and James H. Crossll, (1990) " Reverse Engineering and Design Recovery, Taxonomy" IEEE Software Engineering Vol. 7, No. 1 pp. 13-17.

[7] Gilmore Darid J., Russel L. Winder, and Francoise Petienne, (1991), " User-Centered Requirements for Software Engineering Environment", Series of Computer and System Sciences, Vol. 123.

[8] Baxter, Ira D. and Mehlich, Michael 1997. Reverse Engineering and Forward Engineering. IEEE Service Center/ 445 Hoes Lane/ P.O. Box 1331/ Piscataway, USA.

[9] Faisal bin Bashir and RehanHameed, (1998). Reverse Engineering for Program Understanding: A Case Study. Proceeding of FAST IEEE Student Conference on Computer Science and Information Technology.

[10] Rugaber Spencer (1994). White Paper on Reverse Engineering. Internet Address: http://www.cc.gatech.edu/reverse/tithe.html

[11] Shrivastava Chandra (1996). Reverse Engineering Low-Level Design Patterns from Object-Oriented Code. Louisiana State University, Ph.D. Thesis UMI Microform 9628317, A Bell & Howell Company, http://www.umi.com