



Detection of Reusable Components in object Oriented Programming Using Quality Metrics

Er.NehaBudhija¹,
*M.Tech. (CSE), IGCE
India

Er. Bhupinder Singh²,
Asstt. Prof(CSE),IGCE
India

Er. Satinder Pal Ahuja³
Associate Prof& HOD(CSE&IT),IGCE
India

Abstract- In computer science and software engineering, reusability is the likelihood a segment of source code that can be used again and again to add new functionalities with slight or no modification. Software reuse is the process of reusing the existing software components from the system rather than developing the components newly. Reusable modules and classes reduce implementation time, increase the likelihood that prior testing and reuse has eliminated bugs and localizes code modifications when a change in implementation is required. Reusability is a set of guidelines to help reuser to judge the quality of the component that is to be reused. It is necessary to achieve how much effectively the software component is reusable. To achieve this, the component identification is mandatory. The quality of a reuse system is highly dependent on the quality of the components. Our approach for identifying and qualifying of reusable software components is based on coupling, inheritance, external dependency, and polymorphism and reuse index of the component.

Keywords: Software reuse, reusable components, reuse design, reusability, quality metrics.

I. Introduction

Software has become critical to advancement in almost all areas of human endeavor. The art of programming only is no longer sufficient to construct large programs. There are serious problems in the cost, timeliness, maintenance and quality of many software products. Software engineering has the objective of solving these problems by producing good quality, maintainable software, on time, within budget. To achieve this objective, we have to focus in a disciplined manner on the concept of reusability to develop the product. Software reuse has been widely studied and researched over the past four decades. Software reuse, the use of existing software artifacts or knowledge to build new systems, is pursued to realize benefits such as improved software quality, productivity, or reliability [1].

Component-based software engineering (CBSE) has been a direct result of advances in software reuse. Designing software components for future reuse has been an important area in software engineering. Various characteristics, desired properties and design principles for CBSE have been studied and analyzed. Computing environments are evolving to distributed systems where Object-oriented development had not provided extensive reuse. Component-based software architecture is a high level abstraction of a system. It has architectural elements: components which provide functionality, connectors which describe interactions and configuration which represents the topology of connections between components. This abstraction provides many advantages in the software life cycle like better abstraction capabilities, better flexibility for evolution and maintenance, better reusability as compared to object oriented paradigm. Hence, it is important to extract component based architecture from an object oriented system.

Thus, this paper presents a tool and process for identifying components using existing dependencies among classes. We have evaluated our approach on small java program. The rest of the paper is organized as follows: In section 2 we propose a tool and process for identifying components. Section 3 presents the implementation & Result. Conclusion is provided in section 4. Section 5 proposes idea for future.

II. New Implemented DROOP Tool

The tool DROOP stands for Detection of Reusable components in Object Oriented Programming. Different steps have been identified to produce a component based architectural view from an object-oriented application. Our approach shows overall approach for producing components:

Steps to identify the reusable components:

i) *Identify external dependencies in existing object oriented system:* - Our process is based on the identification of source code entities and the level of dependencies between different classes or packages classes with each other. This metric is also known as cohesion in object oriented paradigm.

ii) *Identify classes:* - A class is a group of various methods and other functionality collaborating to provide a purpose to application. The tool is going to identified total number of classes in an object oriented project.

iii) *Identify the provided and required interfaces:* - Identified group of classes working together will form components. It is also need to identify required and provided interfaces to describe how they bind together. A tool is being developed tool

which will accept the user input of an existing java source code and then generates dependencies. The tool analyzes data represented through these dependencies.



Fig2.1: DROOP TOOL

iv) *Identify polymorphism level in object oriented systems*: - Polymorphism refers to a single function or multi-functioning operator performing in different ways. This parameter is an important parameter in software reusability because it allows routines to use variables of different types at different times. DROOP tool is detecting these useful components in an efficient manner.

v) *Identification of Coupling in systems*: coupling in object oriented systems is the dependency in classes among each other. It is an important parameter to consider in identification of reusable components.

vi) *Identification of Inheritance in object oriented systems*: - Inheritance is other important parameter which plays an important role in object oriented paradigm. It is the detection of classes which are derived from one of more different base classes. These base classed can further based on other base classes. The tool efficiently detects this dependency in object oriented systems.

vii) *Calculation of Reusability Index and value*:- the main aim of this research work is the identification of reusable components. So , the droop generate reusability index and finally a reusability value on the basis of various above parameters count value. There values are efficiently calculated using formula given below

```

/*
 *Reusability Index per class
 */
RUI = RUI + ( poly_count * Config.PW + inheritance_count * Config.IW - coupling_count * Config.CW );
    
```

Fig 2.2: Reusability Index

```

/*
 *Project Reusability value
 */
double PRV = ( RUI - (Config.EW * external_count) ) /class_count;
return PRV;
    
```

Fig 2.3: Project Reusability Value

III. IMPLEMENTATION & RESULTS

We implement this method using Java NETBEANS & Java Swings. In DROOP, we implemented more than ten java files to provide a better GUI tool for reusability testing. The classes are designed through core and advance features of java in Netbeans IDE. The tool calculates all components count value, such as coupling count, polymorphism count, classes count, interfaces count, external dependency count etc. After that it is responsible for calculation of Project reusability value at the end. Finally, a report is generated in the form of excel sheet which consist of weightage count of each component and the value of ReThis method analysis to be more efficient than older developed methods and can be easily used for finding out reusable components from the existing code.

IV. CONCLUSION

In this study, we have implemented a component identification method that considers coupling, interface, inheritance, external dependency and polymorphism. We have developed a tool which accepts object oriented java source code and migrates into component based system. The tool identifies, inheritance, coupling, interfaces, external dependency. We have developed this tool for migration from object oriented system to component based system. Java source code is input to our tool for parsing. It shows inheritance, coupling, polymorphism, external dependency from java source code. This method analysis to be more efficient than older developed methods and can be easily used for finding out reusable components from the existing code.

The accomplished research is related to the understanding of the code written in the java. The designed system discovers suitable reusable components to assist developers. The newly designed tool is fairly efficient to extract the required reusable components. Software developers can be assisted by providing them automatic functionality of reusable components searching.

IV. FUTURE SCOPE

Our future work will focus on defining cohesiveness between components being identified by the tool. Also it is hoped that in future, we will be able to expand the list of reusable components and some new enhanced metrics which will provide more complete measures. Moreover, cyclomatic complexity and volume regularity reuse components can be added for improvement of the tool. Finally, further evaluation on larger and more complex programs is needed to assess how methodology scales to deal with real industrial scale. This tool can be design online for the online detection of reusable component for better service. In future it can be extended to analyze or evaluate various other programming language projects. The GUI (graphical user interface) can be more interactive for better understanding of reuser. The next step is to apply the proposed process model on large scale systems to analyze the benefits and limitations.

In short, there is still a lot that can be done in the field of reusability for software engineering environment.

REFERENCES

1. W. B. Frakes and K. C. Kang, "Software reuse research: status1 and future," IEEE Transactions on Software Engineering, vol. 31, pp. 529-536, 2005.
2. Suresh Chand Gupta, Prof Ashok Kumar "Smart Environment for Component Reuse" Volume 2, Issue 2, February 2012, International Journal of Advanced Research in Computer Science and Software Engineering.
3. "A Strategy to identify components using clustering approach for Component reusability ", NMDJubairBasha and Dr. Chandra Mohan, CS & IT – CSCP 2012.
4. James F Peters, Witold Pedrycz, "Software Engineering, An Engineering Approach", Wiley India Private Limited, 2007.
5. Frakes, W.B. and Kyo Kang (2005) "Software Reuse Research: Status and Future", IEEE Trans. Software Engineering, vol. 31, issue 7, July 2005, pp. 529 - 536.
6. Parvinder Singh and Hardeep Singh (2005) "Critical Suggestive Evaluation of CK METRIC" ,Proc.of9th Pacific Asia Conference on Information Technology (PACIS-2005), Bangkok, Thailand, July 7 –10,2005.
7. Parvinder Singh Sandhu and Hardeep Singh, "Software Reusability Model for Procedure Based Domain-Specific Software Components", International Journal of Software Engineering & Knowledge Engineering (IJSEKE), Vol. 18, No. 7, 2008.
8. Parvinder S. Sandhu, Parvinder Pal Singh, hardeep Singh, "Reusability Evaluation With Machine Learning Techniques", WSEAS TRANSACTIONS on COMPUTERS, issue 9, Volume 6, September 2007.
9. Arun Sharma, Rajesh Kumar and P .S. Grover, "A Critical Survey of reusability aspects for component-based systems", Proceedings of World Academy of Science, Engineering & Technology, Vol. 21, Jan 2007.
10. Xichen Wang, Luzi Wang: Software Reuse and Distributed Object Technology, IEEE Transactions on Software Engineering, 2011.
11. Parvinder Singh Sandhu and Hardeep Singh, "A Reusability Evaluation Model for OO-Based Software Components", International Journal of Computer Science, vol. 1, no. 4, 2006, pp. 259-264.
12. Parvinder Singh Sandhu and Hardeep Singh, "Automatic Quality Appraisal of Domain-Specific Reusable Software Components", Journal of Electronics & Computer Science, vol. 8, no. 1, June 2006, pp. 1-8.