Object oriented Software Visualization for C++ Programming

Ashok Kumar Behera
Bhilai Institute of Technology, Durg (CG)
ashokctc15@gmail.com

Himabindu Maringanti
North Orissa University, Baripada
profhbnou2012@gmail.com

ABSTRACT

It is a costly process for maintaining the pre defined software of the organization. The process of maintenance depends upon the available documentation. *It depends upon documentation. The documentation of software may not be available in industries for further usability of software. To maintain the software, visualization tools are highly required by the industry and research centers. In this paper, I am going to discus the the performance of our Software Visualization Tool, which focuses the class diagram with attributes and methods of the class in object oriented programming. We design the common algorithm for visualization of object oriented program. The aim of our tool is to provide the simplified design of object oriented source code, specifically c++, which can be easily maintainable and usable.*

Keywords: Software visualization, re-engineering, reverse engineering.

1. INTRODUCTION

All the software needs to be maintained from time to time due to new requirement and changes. When the technology or language of the software system is obsolete, it becomes a legacy system yet it is still in use to serve users' needs. Maintaining a legacy system needs a lot of software engineers' time and effort especially when there is no document available or the existing documents are not updated. Software reverse engineering is most costly activity in software development to extract the document from the legacy system. Different tools have been proposed to understand the software system. The application of different visualization techniques [3] has the potential to provide the solution for complex system. The success of the tool depends upon, how it fulfills the need of user.

The tools are widely necessary for the industry for maintenance of the software. The task of tool is to identify the design from the coding of the system. This assists the user of software to understand better about the system.

This paper presents the software visualization support[10] to simplify the complexity of program to lower level. This tool is designed in C++. This provides the the program changes, information that required to maintain the program is specifically changed. The aim of the tool is to provide the design of the program with respect to class and lower up to attributes along with its methods. This will help to re design and maintain the legacy system easily. The visualization of the program is of two types, i.e. dynamic and static. In dynamic visualization the program behavior is to be traced at the time of execution. The behavior of the program varies with respect to change in the argument. Hence the conclusion of visualization will vary in each step. The better approach to find the accurate result is static visualization rather dynamic visualization. In static visualization the source coded of the program is visualized.

2. VISUALIZATION TOOL

Visualization tool

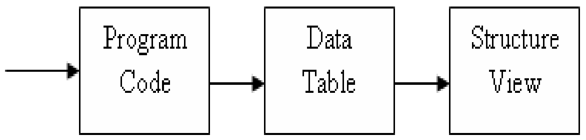


Fig 1: Reference model

Program Code is the source code of input system. Here we are considering C++ programming as input, from which design would be extracted.

Data Table:- Retrieved data like attributes, methods, classes along with relations. Data table extracted from source code through SV-tool are stored in storage media.

Structure View:-Representation of relations. The relation is building in pictorial form, it is class diagram, inheritance and properties present in classes.

4.1 Design of the Tool

Our focus is to simplify the design of complex code, which can be easily maintainable and user-friendly. The main focus is

(a) Need :- The requirement of tool to user. It is designed for research and education purpose.
(b) User :- The users of this tool are research scholar, student and instructors of the institutes.
(c) Representation :- We represent the tool very simple and user friendly. It provides the information on menu based.

Firstly, the program is passed through analyzing tool, which provides the required data necessary for retrieval. The retrieved data is stored in a file for future enhancement. In the next step the information available in the file is represented in graphical format, which can be accessed by the user.

4.2 Methodology of the Tool

The methodology is designed in two parts; one is to extract class along with its attributes and methods. Second part is to design the extracted information in menu based format. This method provides variables and methods of class.

4.3 Architecture of the Tool

This tool provides the sequential architecture, which is of two stages. Stage1 is of metamodel and stage 2 is of visualization. Information will pass through bridge from metamodel to visualization as source code.

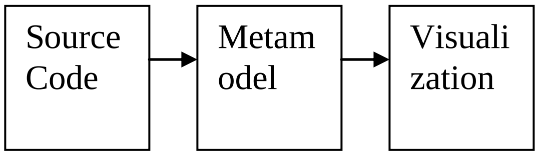


Fig.2 Architecture of the Tool

Source Code is the program written in C++, which have no documentation and required for maintenance. This required visualization of the source code. This pass to the metamodel for tokenize.

Metamodel[7] uses the algorithm shown in figure 2. This is the base model that tokenizes the source code, i.e. the C++ program and stored in a file. All information regarding the visualization of source code are stores in the specified class having attributes as variables and methods. All the information extracted through metamodel are stored in structural way for better representation. This model used for static code. The stored file is then passing for graphical representation.

Visualization is the representation of the design of the code in pictorial form, which can be easily understood by the developer or the user. It includes class diagram along with the behavior of the attributes and methods through inheritance.

4.4 Visualization Engine

The visualization engine [6,7] of SV tool is the user interactive tool itself. It provides the real work of visualization of object oriented program, specifically C++ program. The visualization for the objected oriented program shows the detail of class and its relationship with the other class. The detail of class along with members is shown in figure 3 and figure 8. Both the figures provide the class name along with access-specifier.

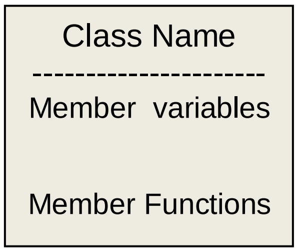


Fig.3 Class diagram

Here in figure 3 the class name along with the properties are given. The properties include the attributes and methods used in class. This tool extracts all classes available in the program. These classes will be displayed and stored in a separate file which can be utilized later. This tool also provides relation between the classes. How the properties are inherited through access-specifier, along with the attributes and methods specified in the class. It shows all types of inheritance. From the inheritance we can extract the properties used by the classes. The sample example is given in figure 4. The detail of inheritance single and multiple are shown in figure 5 and figure 6.

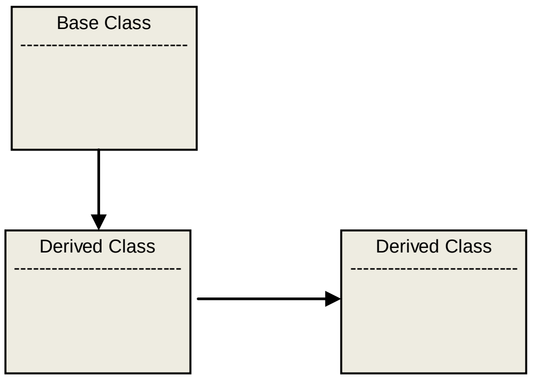


Fig.4 Inheritance of classes

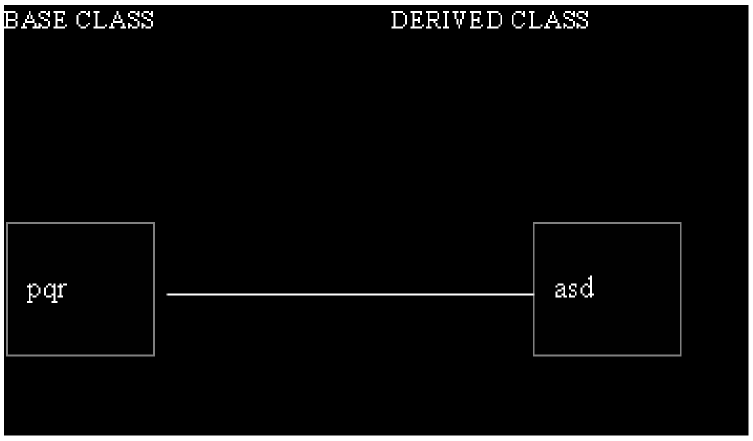


Fig 5 Single level inheritance

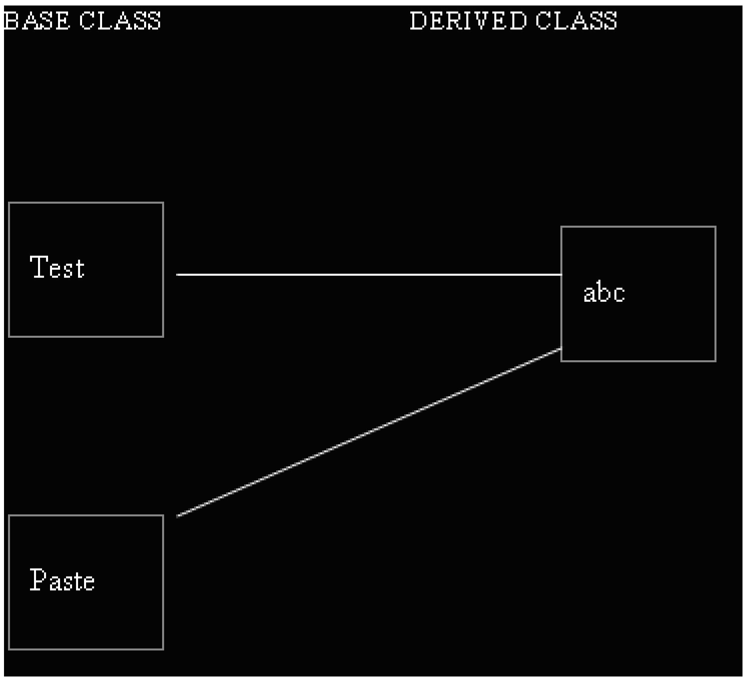
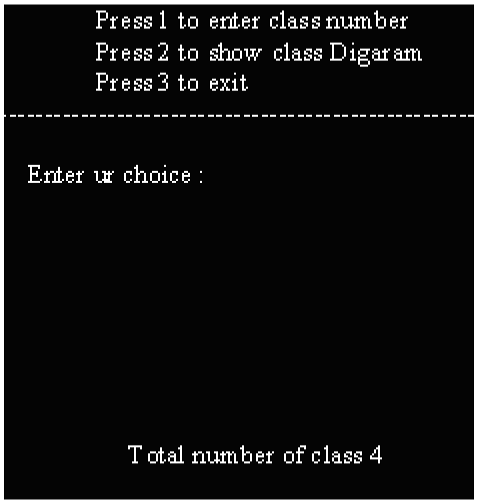
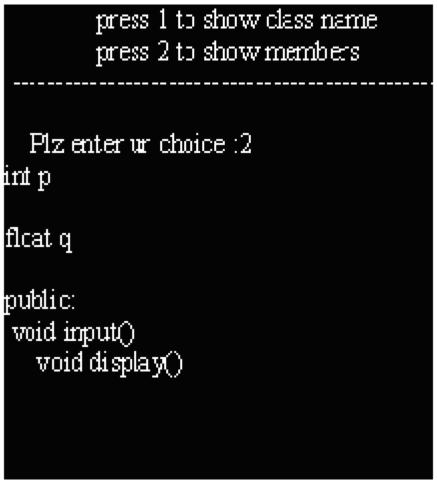


Fig 6 Multiple inheritance



```
        Press 1 to enter class number
        Press 2 to show class Digaram
        Press 3 to exit
------------------------------------------------

  Enter ur choice :





           Total number of class 4
```

Fig 7 Number of classes.



```
        press 1 to show class name
        press 2 to show members
 ---------------------------------------------

  Plz enter ur choice :2
int p

float q

public:
 void input()
   void display()
```

Fig 8 Class detail

The actual single inheritance is shown in fig 5, where base class is pqr and derived class is asd. The total class present in the program shown in fig 7. The multiple inheritances is shown in figure 6, where there are two base classes taste, paste and derived class is abc.

5. CONCLUSION

This paper presents a software visualization of object oriented program, with focus on the inheritance of properties. This provides actual design of program, flow of data and methods from one class to other.

This paper explained the concept of the tool and the techniques used in it. The target of these systems is to provide multiple object-oriented environments. This is mostly helpful for research students and industries used C++ programs. As per the testing concern, it can extract small and medium scale programs and provide the required output. We will further enhance this tool for large scale programs.

References

[1] Eden,A.H., Gasparis,E., Nicholson,J., Razman, R.: Modeling and visualizing object-oriented program with Codecharts. Springer. 2013, Vol 43.

[2] Anslow C., Noble J., Marshall S., Tempero E.: Towards End-User Web Software Visualization. IEEE Symposium on Visual Languages and Human - Centric Computing, 2008 pp 256 – 257.

[3] Michele Lanza: Code Crawler- Lessons learned in building a software visualization tool. IEEE Conf. on Software Maintenance & Reengineering, 2003(0-7696-1902-4). pp 409 – 418.

[4] Stephane Ducasse, Michele Lanza: The class blueprint: Visually Supporting the Understanding of Classes. IEEE Journal 2005, Vol-31, Issue 1. pp 75 – 90.

[5] Thorsten Schater, Mira Mezini: Towards more flexibility in software visualization Tools. IEEE Conf.on Visualizing Software for Understanding & Analysis, 2005. (0-7803-9540-9) pp 1 – 6.

[6] R. Ian Bull and Margaret-Anne Storey, Jean-Marie Favre: An architecture to support model driven software visualization. IEEE 2006 (0-7695-2601-2) pp 100 – 106.

[7] Jonathan I Maletic, Aridrian Marrcus, Michael L. Collard: A Task Oriented view of software visualization. IEEE Conf. on Visualizing Software for Understanding & Analysis, 2002 (0-7695-1662-9) pp 32 – 40.

[8] Michael P. Smith and Malcolm Munro: Runtime visualization of object oriented software. IEEE Conf. on Visualizing Software for Understanding & Analysis, 2002 (0-7695-1662-9) pp 81 – 89.

[9] Anslow C., Noble J., Marshall S., Tempero E.: Towards End-User Web Software Visualization. IEEE Symposium on Visual Languages and Human-Centric Computing. 2008(1-4244-2528-0) pp 256 – 257.

[10] Blaine Price, Ronald Backer, Ian Small: An introduction to Software VisualizationBook with ISBN: 0-262-19395-7.

[11] Mariam Sensalire and Patrick Ogao: Visualizing object oriented software: Towards a point of reference for developing tools for industry. IEEE conference on visualizing software for understand and analyze 2007 (1-4244-0600-5) pp 26 – 29.

[12] Craig Anslow, Stuart Marshall, James Noble, Robert Biddle: Software visualization tool for component reuse Book with ISBN: 0-262-19395-7.

[13] Sensalire M., Ogao P., Telea A: Evaluation of Software Visualization Tools: Lessons Learned. I E E E Conf. on Visualizing Software for Understanding and Analysis, 2009, pp 19 – 26.