ISSN(Online): 2320-9801
ISSN (Print): 2320-9798

International Journal of Innovative Research in Computer and Communication Engineering

(An ISO 3297: 2007 Certified Organization)

Vol. 3, Issue 10, October 2015

An Approach for Extracting Class diagrams from Legacy C++ Code

Ati Jain

Assistant Professor, Dept. of C.S., Swami Vivekanand College of Engineering, Indore, India

ABSTRACT: The research work presented here includes reengineering needs and process framework. It also includes various works that has been done in the field of extraction of class diagram information from C++ source code. Applying concept of parsing the source code keywords can be identified with certain constraints and based on these class diagrams information can be identified. Modules can be made to find the no. of classes, their attributes and its methods. Along with these, class relationships are also to be found out. Current status of software reverse engineering is at very high level. Since the movement of technologies are growing rapidly in such a manner that such kinds of works are to be grown up. Currently, many of the software industries follow the reverse engineering methodology to save cost and time needed to apply new methods. This work shows an approach to extract class diagrams from legacy C++ code in an effective manner.

KEYWORDS: legacy code, reengineering, design patterns, slicing, documentation

I. Introduction

The cost of re-engineering a system is generally less than developing a new system [1]. Sometimes what is required is, add some functionalities, change some policies, change the structure of system without changing functionalities, changing the architecture of the system to add some non functional requirements and for making these changes, it is worthless to develop a new system. But still it is a matter of debate that whether to go for forward engineering or for re-engineering.

Reengineering is an observation, analysis and transformation of existing system into new form, which is more manageable, improved in their functionality so that it can withstand the upcoming technology. Software reengineering is any activity for better understanding of software and/or amends the software itself. It includes source code, design, and documentation so that to improve the understand ability, readability and structure, without changing any of its functionalities. This definition classifies software reengineering into two sets of activities. The first one comprises of activities that support program understanding, such as exploring, understanding, and reverse engineering. The second includes activities targets towards software evolution, such restructuring, re-documentation, and re-modularization [2].

Nowadays software systems are growing fast so every time some changes is required in software. Some legacy software systems are available in many languages. To make any change and also maintenance of these software systems are very difficult due to absence of proper requirement and design documentations. Many legacy software products are difficult to comprehend with complex control structure and unthoughtful variable names.

The focus of this research work is intended to develop a Re-engineering method to automate the extraction of business rules from source code. “A business rule specifies or restraints one aspect of business that is intended to verify business structure or determine the behaviour of our business”. Extracted business rules will be classified as structural, behavioural and constraint rules. In this, a program can be taken as input and then domain variables are identified. The available tools left the task of identification of variables on the maintainer. And good approximation of variable is necessary of extraction of required business rule. The identified variable will be used for program slicing. The output of program slicing will be sliced segments, which will contain the business rule. These sliced segments will be than given to presentation tool to present the rule in different views so that different stakeholder can easily understand the rules.

II. Objective

Reading a code is much difficult task than writing a code (program). As a result, extracting the information from a system, it may be structure, architecture, design, requirements of system, remains a challenging job. The task of a

software re-engineer is to bridge the gap between operational levels at which a program is and the conceptual level, at which a business system works. Re-engineering helps in better understanding of system even if documentation is not available. Re-engineering a system is generally less expensive than developing a new system. A re-engineering method can help in extraction of business rules buried in source code of legacy system. The extracted business rule must be at high level of abstraction so that every stakeholder of the system could understand the system. They need a transparency so that they can well understand the processing of the system.
Reverse-engineering can help maintainers understanding a complex system by retrieving models and documentation from a program. This is the process defined as re-documentation in. UML is a good target language for the reverse engineering of models since it is largely used and offers different diagrams [3].

"Objective of this work is to automate the extraction of information from legacy source code, that information extracted represents the actual design requirements." This will help in re-engineering of the legacy system. The project activity basically consists of an implementation of method for extraction of information related to class diagrams from legacy source code. The main task is to identify the classes and their relationships, which can be association, aggregation, inheritance, composition, realization, dependency, specialization, generalization and multiplicities. Finding out the code segment in program where classes and its related information are buried. For better understanding of the stake holder output should be presented in different views.

This research work is to design and develop Information Extractor system that parses the legacy code in order to get information about the class diagram that can be present in the code. Various other systems previously defined for extracting information, totally based on the gaining knowledge about class diagrams. This system is the one of the method to improve the quality of work [1].

III. METHODOLOGY

Reverse Engineering Procedure in this research work is applied as:

A. *Collect information*

The initial step is collect or ask user to input any C++ code for which class diagram is to be generated.

B. *Record variables.*

The program structure is must analysed to identify variables that are present in CHAR, FLOAT and INTEGER type, along with these there are number of other way of representation of variables with different return types [2][11].

First of all, make a parser which can parse the whole code for searching for the keywords with variable return type. As soon as the keywords are finded out check whether it is a variable declaration or any other type. If it found to be variable declaration store the respective variables with their respective return types.

C .*Record functions (methods).*

For finding various functions in the program, record the total number of functions that are present in the program. Recognize all the functions include various data types. This step is to record the processing associated with each functions.

D. *Identify Class and class dependencies.*

Recognize all the classes that are present in the source code along with the dependencies between various classes as well [2].

E. *Record relationship Association*

Association represents the ability of one instance to send a message to another instance.

1. Create two char arrays name as common1[] and common 2[], first for callee class and second for called class
2. Now we use predefined array in which classes name are stores named as classname[]
3. Parse the Input strings in the class and store this in two character pointers named 'callee' and 'called'

```
while(inputString!=EOF)
      {
       for(i=0;i<n;i++)        //to check each entry in the classname table
```

ISSN(Online): 2320-9801
ISSN (Print): 2320-9798

International Journal of Innovative Research in Computer and Communication Engineering

(An ISO 3297: 2007 Certified Organization)

Vol. 3, Issue 10, October 2015

```
                for(j=0;j<n;j++)
                {
                if(callee==classname1[i][j])
                {
                      common1[]=callee;  //store callee class name into common1
                      }
                      if(called==classname1[i][j])
                       {
                      common2[]=called;  //store called class name into common2
                       }
                      } }
      printf("The class %s is associated with class %s" ,common1,  common2);
```

F. *Record Relationship Aggregation*

Aggregation is the typical whole/part relationship.

1. Create an array classnames[][] that stores the names of the classes.(*/ it stores all the names of the class that are present in the Input source code to be scanned.*/)
2. In the input code scan each and every character by storing it in the character variable

```
If (ch='c')
        {
        If (ch='l')
                {
                If (ch='a')
                        {
                        If (ch='s')
                                {
                                    If (ch='s')
                                        {
                                                If (ch='{')
                                                        {
                                                                then
```

3. Search for a pointer object within the scope.
4. While ch !='}'(End of class scope) call the function end_of_class() make a new character array searchname[],that stores the object from class into the input code by comparing and searching it with the classname[][] array   // store pointer's parent class' name into searchname

```
for(j=0; j<number_of_classes; j++)
 {
        if(searchname[]==classname[0][j])
    {
        goto step 6;
    }
}
```

Class has the aggregation with the class name with the input scan code

G. *Record relationship Generalization/Inheritence*

Inheritance is a way to form new classes (instances of which are called objects) using classes that have already been defined. The new class, known as derived class**.**

H. *Record Relationship Realization*

A realization relationship is a relationship between two model elements, in which one model element (the client) realizes (implements or executes) the behavior that the other model element (the supplier) specifies.

1. Find the existence of inheritence in the classes (for eg. by identifying the presence of : symbol).

ISSN(Online): 2320-9801
ISSN (Print): 2320-9798

International Journal of Innovative Research in Computer and Communication Engineering

(An ISO 3297: 2007 Certified Organization)

Vol. 3, Issue 10, October 2015

```
    if(inheritence found) continue.
 2. Create one array named as char method1[]      //to store the names of the overriden methods
 3. Now we use predefined array used in previous code named as methodnames[]
 4. Parse the Input string  in the class and it is stored in char variable char check
 5. Create an integer variable realization count int rc=0; //to keep count of number of methods that are overridden.
      while(inputString!=EOF)
      {
              for(i=0;i<n;i++)
              for(j=0;j<n;j++)   //searching the table for overridden methods
              {
                        if(check==methodnames[i][j])
                         {
      Method1=check;
      rc++;
                         }
    }
    }
      printf("The realization count is %d",rc);
      printf("The method which is overridden is %s",method1);
```

I. *Record Relationship Dependency*

Dependency is a weaker form of relationship which indicates that one class depends on another because it uses it at some point of time.

```
1.  Create one arrays name as char dependency1[]
2.  Now use predefined array as used in previous code named as Objectnames[]
3.  Parse the Input string  in the class and it is stored in char variable char check
4.  Create an character array next[] ;
    while(inputString!=EOF)  {
    for(i=0;i<n;i++){ //searching for dependencies by looking for matches in the table
         if(check==Objectnames[i])
         {
              if(Inputstring=='.')//to check an object call
              {
                        while(inputString!='\0')
                        {
                                  Next[i]=InputString;    //store method name
                                  }
                        }
              }
    }
    for(i=0;i<n;i++)
              {
              for(j=0;j<n;j++)
              {
                        if(Next==classname[i][j])
                                       goto print statement;
                                       }
                                  }
              }
printf("The Dependency Attribute is %s",next);
```

ISSN(Online): 2320-9801
ISSN (Print): 2320-9798

International Journal of Innovative Research in Computer and Communication Engineering

(An ISO 3297: 2007 Certified Organization)

Vol. 3, Issue 10, October 2015

IV. CONCLUSION

The automation of re-engineering process is not an easy task and maintainers are not in favor of a tool which directly re engineers a system. As sometimes they are interested in extracting little information from the system in the absence of documentation. In this paper work a novel approach for finding class diagrams information by reverse engineering C++ code is generating a parser like tool.

REFERENCES

1. Chih-Wei Lu, William C. Chu, Chih-Hung Chang, Yeh-Ching Chung, Xiaodong Liu, Hongji Yang: ―Reverse engineering, Handbook of Software Engineering and Knowledge Engineering, 2002.
2. S. R. Tilley, H. A. Muller, M. J. Whitney, and K. Wong. Domain-retargetable reverse engineering. In Proceedings of the 1993 International Conference on Software Maintenance (CSM '93), (Montreal, Quebec; September 27-30, 1993), pages 142{151. IEEE Computer Society Press (Order Number 4600- 02), September 1993.
3. J Shao and C J Pound," Extracting business rules from information system ", IEEE, volume 3 Nov2005,
4. H. Huang, W.T. Tsai, S. Subramanian, "Generalized Program Slicing for Software Maintenance", to appear in Proceedings of SEKE'96.
5. H. Huang, W. T. Tsai, S. Bhattacharya, X. P. Chen, Y. Wang and J. Sun "Business Rule Extraction from Legacy Code", TR 9.5-34, Department of Computer Science, University of Minnesota, Minneapolis, MN 55455, 1995.
6. Chia-Chu Chiang; Bayrak, C. ―Legacy software Modernization‖ Systems, Man and Cybernetics, 2006. SMC apos; 06. IEEE International Conference on Volume 2, Issue , 8-11 Oct. 2006 Page(s):1304 – 1309.
7. Yang, H; Luker, P and Chu, W, "Code Understanding Through Program Transformation for Reusable Component Identification", 5th International Workshop on Program Comprehension (IEEE IWPC 97) 148-157, 1997.
8. J. K. Joiner, W. T. Tsai, X. P. Chen, *S* . Subramanian, J. Sun and H. Gandamaneni, "Data-Centered Program Understanding", Proc. of IEEE Software Maintenance, 1994,
9. Joseph M Scandura ―Converting Legacy Code into Ada University Of Pennysilvania and intelligent Micro Systems, Inc. April 1994.
10. Booch, G, Rumbaugh, J, and Jacobson, I, *The Unified Modeling Language User Guide,* Addison Wesley-Longman, Inc, 1999.
11. X. P. Chen, W. T. Tsai, J. Joiner, H. Gandamaneni and J. Sun, "Automatic Variable Classification for COBOL Programs", Proc. of IEEE COMPSAC, 1994, pp. 432-437.
12. C. L. Ong and W. T. Tsai, "Class and Object Extraction From Imperative Code", Journal of Object-Oriented Programming, March/April 1993, pp. 58-68.

BIOGRAPHY

Ati Jain is an Assistant Professor in the Computer Science Department, Swami Vivekanand College of Engineering, Indore. She received Master of Technology (M.Tech) degree in 2011 from RGPV University, Bhopal, MP, India. Her research interests are Software engineering, object oriented analysis and design.