

THE AGENT BASE CODE COMPREHENSION

A Dissertation submitted by

AROOSHI

to

Department of Science and technology

In partial fulfilment of the Requirement for the

Award of the Degree of

Master of Technology in Computer Science

Under the guidance of

ROHITT SHARMA

(May 2015)

PAC form

ABSTRACT

Program comprehension is an important part of software maintenance, especially when program structure is complex and documentation is unavailable or outdated. This work presents a method of understanding the code aiming at capturing program structure and dependencies between the object, classes, functions and other modules and achieving better system understanding. The system source code first fed into the GCC XML to facilitate the use of wide variety of XML, with the help of GCC XML tool the source code is being converted to XML output which give data in the form of tree and this form of XML is also equivalent to c++ parser. The resultant XML output get converted to XLS file from that internal dependencies are being fetched and sorted manually.

We try to implement that how this work fits in the context of tool supported maintenance and comprehension and report on applying a new methodology on C++ programs. The XLS file give that data in separate file it help to visualize about whole source code its withdraw all the details about the source code in separate file. From the sorted data the dependencies graph are drawn which represent the whole structure f the source code including dependencies between functions, classes, variables and different modules.

CERTIFICATE

This is to certify that Arooshi has completed M.tech dissertation titled “**Agent based code comprehension**” under my guidance and supervision. To the best of my knowledge, the present work is the result of his original investigation and study. No part of the dissertation has ever been submitted for any other degree or diploma. This dissertation is fit for the submission and the partial fulfilment of the conditions for the award of M.tech Computer Science & Engineering.

Date: _____

Signature of Advisor

Name: Rohitt Sharma

ACKNOWLEDGEMENT

“Candle burn itself to give the light” –likewise the Teacher

I shall not be able to accomplish this much of work without the help, support and able guidance of Mr. Rohitt Sharma, with transparent clarity. He has helped me throughout the demonstration with his pool of knowledge, extreme support for being available and giving extra hours for accomplishing the work, I am heartily obliged to him for such a great support. I also heartily want to thanks to other faculties who supported for this work.

DECLARATION

I hereby declare that the dissertation proposal entitled, Agent based code comprehension submitted for the M.Tech Degree is entirely my original work and all ideas and references have been duly acknowledged. It does not contain any work for the award of any other degree or diploma. This dissertation is fit for the submission and the partial fulfilment of the conditions for the award of M.tech Computer Science & Engineering.

Date: _____

Arooshi
Reg No - 11004845

TABLE OF CONTENTS

| | | |
|---|-----------------------------|----|
| Acknowledgement | v | |
| Declaration | vi | |
| CHAPTER 1 | Introduction | 1 |
| 1.1 Software Maintenance Basics | 4 | |
| 1.1.1 Terminology and Definition | 4 | |
| 1.1.2 Essence of Maintenance | 4 | |
| 1.1.3 Need for Maintenance | 6 | |
| 1.1.4 Majority of Maintenance Costs | 6 | |
| 1.1.5 Evolvement of Software | 6 | |
| 1.1.6 Classes of Maintenance | 5 | |
| 1.2 Software Maintenance Key Points | 6 | |
| 1.2.1. Technical Points | 7 | |
| 1.2.2 Finite understanding | 7 | |
| 1.2.3 Testing | 7 | |
| 1.2.4 Impact analysis | 7 | |
| 1.2.5 Maintainability | 8 | |
| 1.2.6 Issues in the Management | 8 | |
| 1.2.7 Adjustment with organizational target | 8 | |
| 1.2.8. Staffing in software maintenance | 8 | |
| 1.2.9 Process | 8 | |
| 1.2.10. Condition of maintenance in Organizational | 8 | |
| 1.2.11. Maintenance Cost Estimation | 8 | |
| 2.1 Maintenance Techniques | 8 | |
| 2.1.2. Maintenance Processes | 9 | |
| 3.1 Techniques for Maintenance | 9 | |
| 3.1.2. Program Comprehension | 10 | |
| 3.1.3. Reengineering | 10 | |
| CHAPTER 2 | Review of Literature | 11 |

| | | |
|---|------------------------------------|----|
| CHAPTER 3 | Present Work | 14 |
| 3.1 Objective of Research | | 14 |
| 3.2 Scope of Research | | 15 |
| 3.3 Algorithm of Proposed Method | | 16 |
| CHAPTER 4 | Result and Discussion | 18 |
| CHAPTER 5 | Conclusion and Future Scope | 27 |
| | List of References | 29 |
| | Appendix | 32 |

LIST OF TABLES

| | | |
|----------------|--|----------|
| Table 1 | Software maintenance categories | 6 |
|----------------|--|----------|

LIST OF FIGURES

| | | |
|-------------------|--|-----------|
| FIGURE 1 | Disintegration Software Maintenance topic | 3 |
| FIGURE 2 | IEEE1219-98 Process Activities of maintenance | 9 |
| FIGURE 3 | Overview how work will be process. | 17 |
| FIGURE 4.1 | Banking system code | 19 |
| FIGURE 4.2 | Banking system code. | 19 |
| FIGURE 4.3 | Process for running GCC XML | 20 |
| FIGURE 4.4 | Output XML file from GCC XML | 21 |
| FIGURE 4.5 | XML view file | 21 |
| FIGURE 4.6 | XLS Account class and its functions | 22 |
| FIGURE 4.7 | XLS Main function | 23 |
| FIGURE 4.8 | XLS files number of variable in class | 23 |

Chapter 1

INTRODUCTION

Software Engineering deals with the three basic area of study and these applications of engineering are development of the software as well as the design and maintenance.

Software engineering standard definitions:

"The application of a systematic, disciplined, quantifiable approach to the development, operation, and maintenance of software"

- "An engineering discipline that is concerned with all aspects of software production."
- "The establishment and use of sound engineering principles in order to economically obtain software that is reliable and works efficiently on real machines"

The below definition use informally:

- As the informal contemporary term for the broad range of activities that were formerly called computer and systems analysis"
- As the broad term for all aspects of the practice of computer programming, as opposed to the theory of computer programming, which is called computer science"

Sub types

Within the field of software engineering:

- Software requirements
- Software maintenance
- Software design
- Software construction
- Software engineering professional practice Software engineering professionalism
- Software configuration management
- Software engineering management
- Software quality
- Software engineering process

- Engineering foundations
- Computing foundations
- Mathematical foundations
- Software engineering models and methods
- Software testing

Some more field of software engineering but have not been define are:

- Mathematics
- General management
- Quality management
- Project management
- Computer engineering
- Systems engineering
- Computer science

Software maintenance is the topic of the proposed research work.

Software Maintenance

After delivery of the product it is the changes in the software to rectify the defect, to upgrade performance or supplementary attributes.

Further sub division of Software maintenance are:

- Fundamentals Software maintenance
- Software maintenance key issues.
- Process of maintenance.
- Maintenance techniques

The efforts which are being done for consequence in the shipment of a software product that content user requirements. Accordingly, there should be changes in the software product that includes in uncovering of defects, change in operating environment operation, and the new surface of user requirements.

The essential chunk of software cycle is maintenance. But, it hasn't the same reorganisation as other stages have, classically. Moreover, historically the software maintenance in most organisations has had lower profile than software development. But

now scenario has been changed as organisations now preserving to focus most on the software development investment. The historically details about the Year 2000 concur showed important focused on software maintenance phase, and the Open Source standard has also showed up the attention to the issues related to software maintenance artefact developed by others. The further sub division or categories of maintenance is being showed in figure one below ^[1].

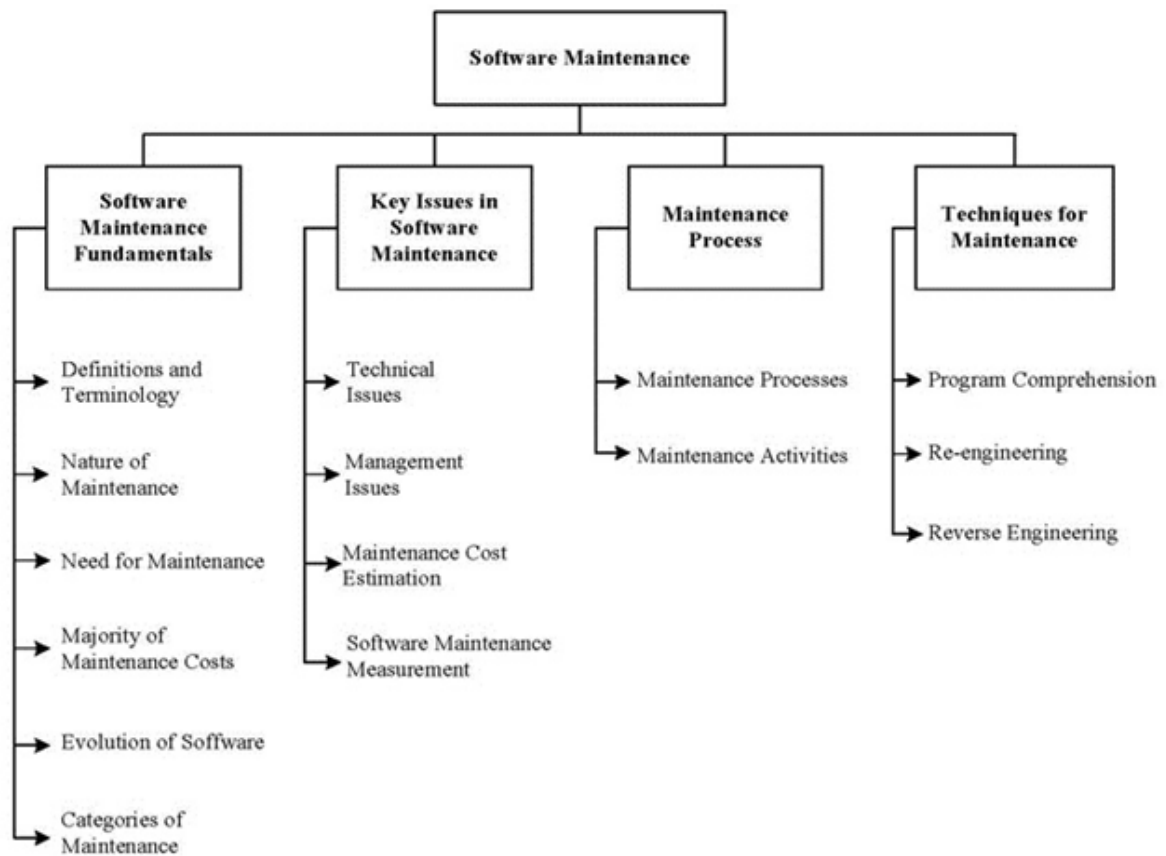


Figure 1: Disintegration Software Maintenance topic.

1.1 Software Maintenance Basics

This part discuss about the approaches and nomenclatures for the maintenance that help to understand the basic and role of maintenance phase.

1.1.1 Terminology and Definition

Software maintenance characterize as the changes in the product of software that incorporate correction of faults, to boost the performance or other aspect or to accustom the product into the altered environment after the delivery of a software product

1.1.2. Essence of Maintenance

Maintenance under software helps to keep the product of software all over its operational life cycle. Appeal for modification are tracked and logged and it determines the brunt of expected change, code and alternative software artefacts, testing, and a new discharged version of the software product. Also, regular supports are handover to users.

The IEEE/EIA exemplifies a maintainer as an organization which executes activities of maintenance. The style basically refers to the individual who will sometimes assign to individuals who perform those actions in contrast with the builders.

From the knowledge of the developers maintainers can attain the insight of the software. The maintainer can reduce maintenance efforts by Contacting with the builder or developer and early and quick involvement by maintainer. Generally, the challenges for maintainer are evolved when they cannot reach or not has moved on to other tasks. The maintenance team should take the product of development, documentation or code and should immediately guide them and sustain them constantly over the software cycle.

1.1.3. Need for Maintenance

To assure that software is preserver to satisfy the user requirements for the maintenance is required. The software which being refined by using any software life cycle model needs the maintenance.(For example, spiral).Because of the curative and non-curative software activities system gets changes. The software maintenance has been performed in the following order:

- Correct faults
- Implement enhancements
- Adapt programs so that different hardware, software, system features, and telecommunications facilities can be used
- Interface with other systems

- Improve the design
- Migrate legacy software
- Retire software

1.1.4. Majority of Maintenance Costs

The financial resources are mostly consumed by the software life cycle maintenance phase. The common anticipation in the maintenance is that it hardly fixes the defects. But, according to the studies it has been showed that over 80% of the software maintenance efforts is used for non-curative actions.

The aspect affecting the technical and non –technical software maintenance is as follow:

- Software novelty
- Software maintenance staff availability
- Application type
- Hardware characteristics
- Quality of software design, construction, documentation and testing.
- Software life span

1.1.5. Evolvement of Software

The work of maintenance is continues developing process .The large software is continues to evolve and never complete. It keeps on grows as it evolve until and unless some activities is being taken in consideration to decrease the complexity.

1.1.6. Classes of Maintenance

It includes four classes, as follows:

- Corrective maintenance: Alteration of the product of software reactively after shipped to correct detected problem.
- Adaptive maintenance: The changes in performed after the delivery of software product to keep software useful in alternated environment.
- Perfective maintenance: In this alternation take place after the delivery as to advance the maintainability or performance.

- Preventive maintenance: In this alternation take place after the delivery of the product as to identify and correct defects in the software.

Adaptive and perfective classifies maintenance enhancements by ISO/IEC 1476. It also groups together correction categories which includes the corrective and preventive maintenance and perfective, adaptive in enchantment categories. as shown in Table 1.

| | Correction | Enhancement |
|-----------|------------|-------------|
| Proactive | Preventive | Perfective |
| Reactive | Corrective | Adaptive |

Table 1: Software maintenance categories

1.2. Software Maintenance Key Points

The key point in software maintenance is requires so as to assure the adequate maintenance of the software. The software maintenance serves management and technical objections for software engineers. Finding the fault in software containing 500k LOC that is not developed by the engineer is a good example.

Some management and technical issues related to software is being showed in the following section:

- Management issues
- Technical issues
- Cost estimation and
- Measures

1.2.1. Technical Points

1.2.2. Finite understanding

By Finite understanding basically means comprehension the software swiftly by software engineer so that he can understand that where to make changes or corrections in software product even when software is not developed by that individual. According to the research it has shown that 40-60% of efforts done by the maintenance phase are dedicated to understanding the software to be altered. Thus, the comprehension is the most important topic of interest in the software engineering, but the comprehension is the most difficult task in the software engineering.

1.2.3. Testing

The testing has significant role in the software as its affect the cost of repeating the full testing which can be more important in terms of time and money. In the maintenance the regression testing is quite significant as to verify the modification.

1.2.4. Impact analysis

In the existing software how to conduct, effective cost and the impact of the change is described by the Impact analysis.

1.2.5. Maintainability

Maintainability is often difficult to achieve to reduce the maintenance cost. The maintenance sub characteristics should be specified controlled and reviewed. Maintainability if the software will be improves if it will be done successful. But it is challenging to accomplish because the alternate characteristics of maintainability are not focused and taken into consideration. The builders are absorbed with many other things and often disrespect the maintainer's needs.

1.2.6. Issues in the Management

1.2.7. Adjustment with organizational target.

The main objective of Organizational is to describe that how to define the return on the investment. The main objective is to deliver within in the budget and in the time that meets the user need. In extension, it may be driven by the requirements to conform user need for software updates and improvements. In both cases, the return on investment is much limited clear, so that the sight at senior management level is regularly of a significant activity consuming important resources with no clear calculable perk for the organization.

1.2.8. Staffing in software maintenance.

Keeping and maintain the software maintenance staff is come under staffing. Maintenance is not related to glamorous work.

1.2.9. Process

Process or the action is the set of methods, actions, transformation and method which is used to develop and preserve the software and the related products.

1.2.10. Condition of maintenance in Organizational.

The objective Organizational conditions are to detect that which organization will be highly liable for the software maintenance. The developing is not importantly appoint to sustain the software.

1.2.11. Maintenance Cost Estimation

It includes the estimating of the cost of the software maintenance. Engineers should well apprehend the different - different classes of the maintenance phase.

2.1 Maintenance Techniques

The Maintenance techniques sub-area accommodates resources and specification used to implement or tool the software maintenance techniques.

2.1.2. Maintenance Processes

The processes of the maintenance are described in the details according to the software maintenance level IEEE 1219 and ISO/IEC 14764.

Below in the figure the process or activities model for the maintenance is being described according to the standards for software maintenance (IEEE1219) begin with the software maintenance. That activities is explained in Figure 2 ^[1].

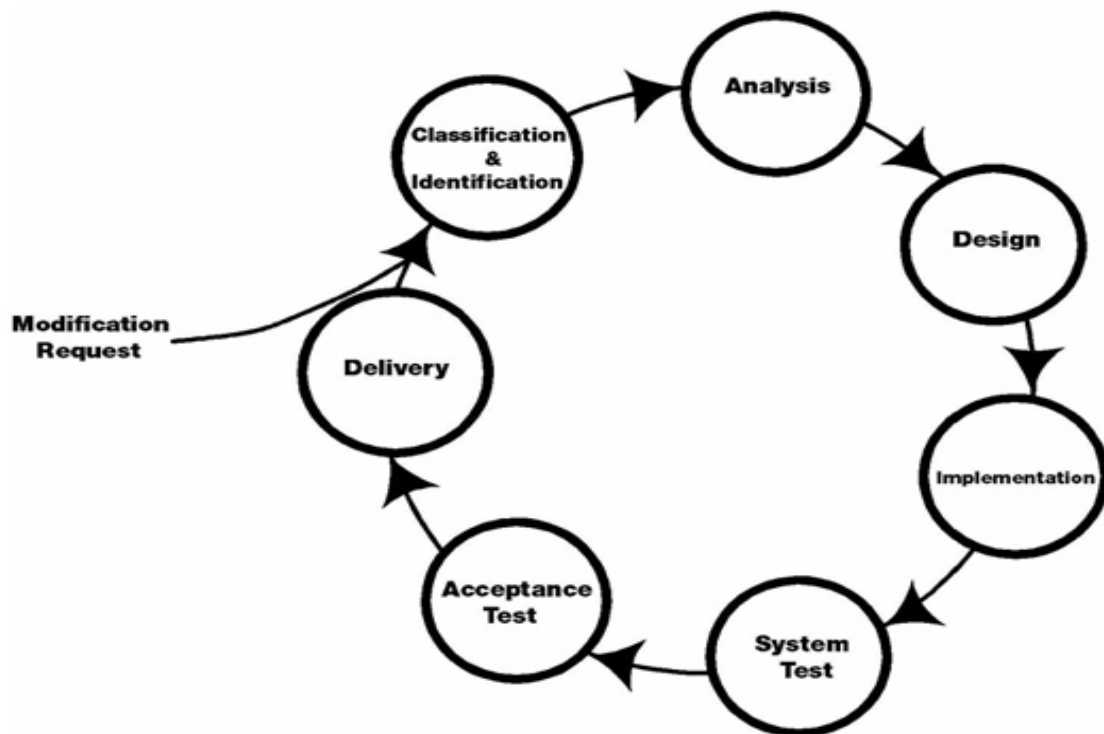


Figure 2 The IEEE1219-98 Process Activities of maintenance

3.1 Techniques for Maintenance

This are describe about the techniques used in the software maintenance and these are the techniques which are been accepted.

3.1.2. Program Comprehension

The program comprehension is the times spend by the programmer, developer or novice in understanding and reading in order to implement the changes. The documentation which is understandable and brief can be help in the program understanding

3.1.3. Reengineering

Alteration or changes in the software to fix up it in the new form is known as the Reengineering. The process of analysis and understanding the component of the software and interrelationship between them so that new software can be created in another or higher form. The reverse engineering process doesn't do convert the software or result in different software it is a passive approach. The call graphs and control flow graph from the code is being generated by the Reverse engineering .The re-documentation and design recovery is another type of reverse engineering. The reverse engineering help to produce the amended version of the software product and it will help to generate the higher and modified version of the product. Reverse engineering help to remove the errors and faults in the software product and it produce the modified and well defined and highly compatible software product.

Maintenance work:

The maintenance work in this proposed method is being done by the GCC XML tool. The GCC XML tool is the XML output addition of GCC. It is the advancement tool that work with programming language it work with the capability to comprehend the code which it work standard equivalent to c++ compiler. The motive of the GCC XML is to bring about an XML description of the c++ program from GCC internal illustration..

GCC-XML was developed by Brad King at Kitware to be used by CABLE, which was developed as part of the NLM Insight Segmentation and Registration Toolkit project.

Chapter 2

REVIEW OF LITERATURE

The agent based code comprehension of c++ program code is the selected topic. As not much work has been done on the program comprehension, few related paper has been done on program comprehension. These related works has shown that how program can be comprehended ,all the following work conclude that how the program comprehension can be achieve and can help to maintainer of software system to maintain the maintenance of system.

Christos Tjortjis ,Loukas Sinos,Paul Layzell (2003), “Facilitating Program Comprehension by Mining Association Rules from source Code”, presented method for mining association from code to frame the program structure and better understanding of the system. The input data is being extracted from the source code and association rule are being applied on it. Association rule is processed the program into groups according to interrelated entities. Entities are clustered together if attributes have common rules. The program is fed into code analyzer to get the input for mining tool. Results show that the method facilitates program comprehension by only use of source code where documentation and reliable documentation are not available. This work facilitates program comprehension and group entities according to their similarities.

Michael L. Collard. Huzefa H. Kagdi. Jonathan I. Maletic, “An XML-Based Lightweight C++ Fact Extractor”, used the lightweight fact extractor which is a vital tool for reverse, reengineering, maintenance, testing and general development of software system. The lightweight utilizes the XML tools to extract the information from c++ source code .The source code is converted into XML .The partial parsing of the source code is done. On low level details this approach cannot be directly addressed .Fact extractor are widely used to support understanding task associated with maintenance, reverse engineering and various other software engineering tasks.

Yiannis Kanellopoulos and Christos Tjortjis, “Data mining Source Code to Facilitate Program Comprehension :Experiments on Clustering Data Retrieved from c++ Programs”, this paper presented the work by using data mining to discover the knowledge about software system and hence facilitate the program comprehension. It discussed how tool supported maintenance and comprehension and leads to apply a new methodology on c++ program. The overall work gives practical insight and guide to maintainer. This work is being done in two ways: it provided the model and associated method to extract the data from c++ source code which is being mined .And audits a proposed framework for clustering to obtain the useful knowledge .The methodology in this paper is evaluated on three open source applications ,results are being assessed and conclusion are presented.

Claire Knight and Malcolm Munro, “Program Comprehension Experiences with GXL; comprehension for comprehension”, in this paper the vitality of the tools has been showed that supports the activities in the program comprehension process. The use of GXL, SORTIE and Graph tools are being used .This papers presented that all the real world applications of program comprehension research allow for a true test of theories and tools.GXL is being used by the author to explore the interoperability, SORTIE used for re-engineering tools. The graph tool supports or includes many features such as reading/writing of four file format: 2dg, .gxl, .gin ,.cll; layout the algorithms, use controlled dragging of nodes and edges, use of various colours for nodes and edges, generation of postscripts out of graph ,generation of JPEG image of the graph, 100% view of graph with a zoom, anonymous setting to obscure real names on graph. Basically this paper highlighted the use of program comprehension for both the analysis of a system to provide recommendation and also to make changes to the statements.

M.-A.D. Storey, K. Wong and H.A. Muller (2000) “How do program understanding tools affect how programmers understand programs”, This paper explored the question about the program understanding that whether the program understanding tools are helping the programmer or they are bringing the change in which way the programmer understand the code. The approaches of understanding the code vary considerably. The approaches of understanding the code should enhance or provide an ease for the programmer to understand the code rather than providing the strategies that not always suitable for the programmer to understand the code. This paper show the observations

from user study which do comparison with three different tools for inspection program code and software structure. Around 30 people used these tools for high level code for understanding task. These tasks need border range of comprehension. This paper describes how the tool helped in comprehension approaches.

Chapter 3

PRESENT WORK

3.1 Objective of research:

The comprehension of the c++ program code is the main objective of study which provides ease of apprehensions. The work presents aims to help maintainer to facilitating the program understanding, and recognise the part of c++ code that have common characteristics. This work comprehends the legacy system so that new system can be easily can be generated from the existing one.

This work focuses on extracting data from c++ code which will be clustered in order to identify correlations among program components.

As the object oriented language, it can be analysed in more details.

Objective of this work is:

- Extract data from c++ code.
- Identifying the relationship and dependency in the code with the help of tools.
- Using the appropriate methodology in producing valid, useful results and knowledge about a software system.

Program comprehension is an important part of not only software maintenance, but also the entire software engineering process, this work is carried out with the aim of well understanding the existing piece code and building the understanding by examine the extracted source code by using various tools for carried out this work.

Extracting the facts dependency and relationships from source code is the main objective of this work. The figure 3 depicts the overview of how the work will be taken in the account.

The objective of the work is to develop the concise way of comprehending the code so that one can understand the legacy system without going through over all code inspection of the legacy system.

The objective of code comprehension is reducing the human effort in understanding the code, saving the time of maintainer and developer in comprehending and building the new system from the existing system. Huge line of codes can be easily understood by the dependency graph instead of going through the line by line of the code. It provide flexibility in understanding of the code and can help to do emendation in the code without generating the errors as dependency graph can help to understand the dependency of function ,method and classes in the code.

3.2 Scope of Research:

The agent based code comprehension technique will help us to develop the flexible documentation of c++ program code. As under the maintenance the problem faced by the maintainer is lack of familiarity and knowledge with code and lack of accurate documentation.

In the absence of expertise, loss of documentation, non availability of the developer, non availability of the manual or when newly recruited person to the particular system who doesn't built that system in all these case, this will leads to the partial understanding of a system and give rise to the increased code complexity and deteriorated the modularity that result in 50-90% of maintainer time to be spend on program comprehension ^[2]. Neither also as there are no explicit guidelines given a programmer understand task, nor there good criteria to decide how to represent knowledge and use for it.

As it is not easy for newly recruited person to understand the code who hasn't built it. It become difficult to go for the inspection of the code and it may sometime leads to the generation of errors and mistakes. Inspection of code doesn't even tell the latent dependencies in the code and it become burdensome for the developer or maintainer to comprehend the code while making emendation in the code. In the loss of documentation or non availability of the developer one has to go for the inspection of the code that again leads to the generation of error and mistakes. Even in the case of non updated documentation, developing the new system from the existing one can leads to development of the inappropriate system that later become problem for the developer and the customer.

And again for the maintainer it will be the problem for correcting the newly build system. Because of the availability of inappropriate code, updating the documentation of the system would also be not correct that further will be the problem for the maintainers.

But this work shall lead to remove all these shortcomings and provides ease of understanding the system. It will lead to save time, reduce and provide ease to make emendations in the code without affecting the other modules of the code.

Also, this work provides an ease of updating the documentation with more flexibility, reliability and this proposed work help to show the hidden dependencies in the system which is generally not being able to trace in documentation, UML and control flow.

A well documented problem faced by maintainers will be overcome by this work and this work helps us to provide the accurate documentation with better understanding of the system.

3.3 Algorithm of proposed:

Step 1: Consider any legacy system code (Banking).

Step 2: By using GCC XML tool convert the legacy system code into XML view file.

Step 3: By using the tool convert the XML view file into XLS file.

Step 4: Sort the data manually according to the calling of function and methods.

Step 5: Draw the dependency Graph through the sorted data of legacy system code.

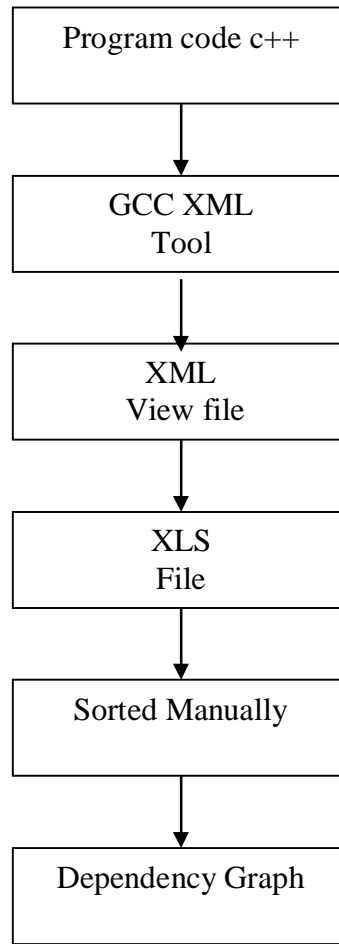


Figure 3- Overview how work will be process.

Chapter 4

RESULT AND DISCUSSION

Research methodology basically means meet the objective of our work so that we can reach to the remainder of our work and gets the expected outcomes or results. Only by using the appropriate research methodology work can lead to get the correct and valid outcomes.

The proper use of techniques and tools should be used in order to get expected outcomes or to reach the remainder of the work. In this work ,the legacy system would be taken in the consideration ,which will be get parsed and give the output of the code in form tree in XML view file by the use of GCC XML tool . The source code is first converted to the XML representation, to facilitate the use of wide variety of XML tools and it will be converted to the XLS (spread sheet) file, which eventually sorted manually according to the dependency among the code.

The proposed method is implemented on banking system. The banking system is about the whole banking process. It includes create account, details of account, withdraw money, deposit money, balance details, show account it show all the details from the depositing of money to withdrawal of money. Account type whether it is a saving account or the current account. Return the account number if user just want to show the account number .Modify the account if user want to change credentials of the account. This banking code is for the banking system that includes all the basic banking process requirement. The banking system includes one class name Account which include functions that are:

Create account (); It is the function to get data from user

Show account () const; It is the function to show data on screen

Modify (); It is the function to add new data

Dep (int); It is the function to accept amount and add to balance

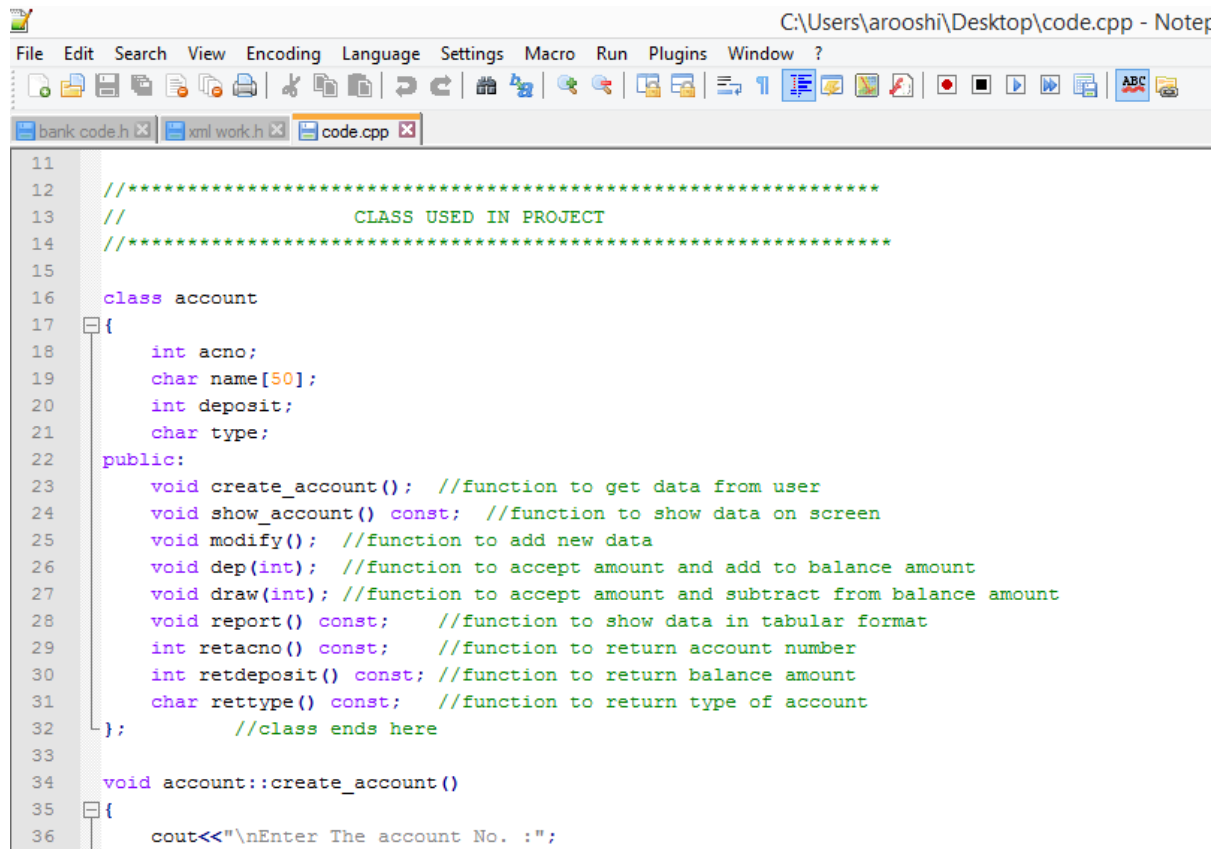
Amount draw (int); It is the function to accept amount and subtract from balance amount

Void report () const; It is the function to show data in tabular format

Int retacno () const; It is the function to return account number

Int retdeposit () const; It is the function to return balance amount

Char rettype () const; It is the function to return type of account



```
11
12 //*****
13 //          CLASS USED IN PROJECT
14 //*****
15
16 class account
17 {
18     int acno;
19     char name[50];
20     int deposit;
21     char type;
22 public:
23     void create_account(); //function to get data from user
24     void show_account() const; //function to show data on screen
25     void modify(); //function to add new data
26     void dep(int); //function to accept amount and add to balance amount
27     void draw(int); //function to accept amount and subtract from balance amount
28     void report() const; //function to show data in tabular format
29     int retacno() const; //function to return account number
30     int retdeposit() const; //function to return balance amount
31     char rettype() const; //function to return type of account
32 }; //class ends here
33
34 void account::create_account()
35 {
36     cout<<\"\\nEnter The account No. :\";
```

Figure 4.1 Banking system code.



```
181 //*****
182 //          function to write in file
183 //*****
184
185 void write_account()
186 {
187     account ac;
188     ofstream outFile;
189     outFile.open("account.dat",ios::binary|ios::app);
190     ac.create_account();
191     outFile.write(reinterpret_cast<char *> (&ac), sizeof(account));
192     outFile.close();
193 }
194
195 //*****
196 //          function to read specific record from file
197 //*****
198
199 void display_sp(int n)
200 {
201     account ac;
202     bool flag=false;
203     ifstream inFile;
204     inFile.open("account.dat",ios::binary);
205     if(!inFile)
```

Figure 4.2 Banking system code.

The legacy system code with the help of GCC XML is been converted in the form of tree XML view file. The GCC XML tool is the development tools that wok with programming language. The purpose of the GCC-XML extension is to generate an XML description of a C++ program from GCC's internal representation. Since XML is easy to parse, other development tools will be able to work with C++ programs without the burden of a complicated C++ parser.

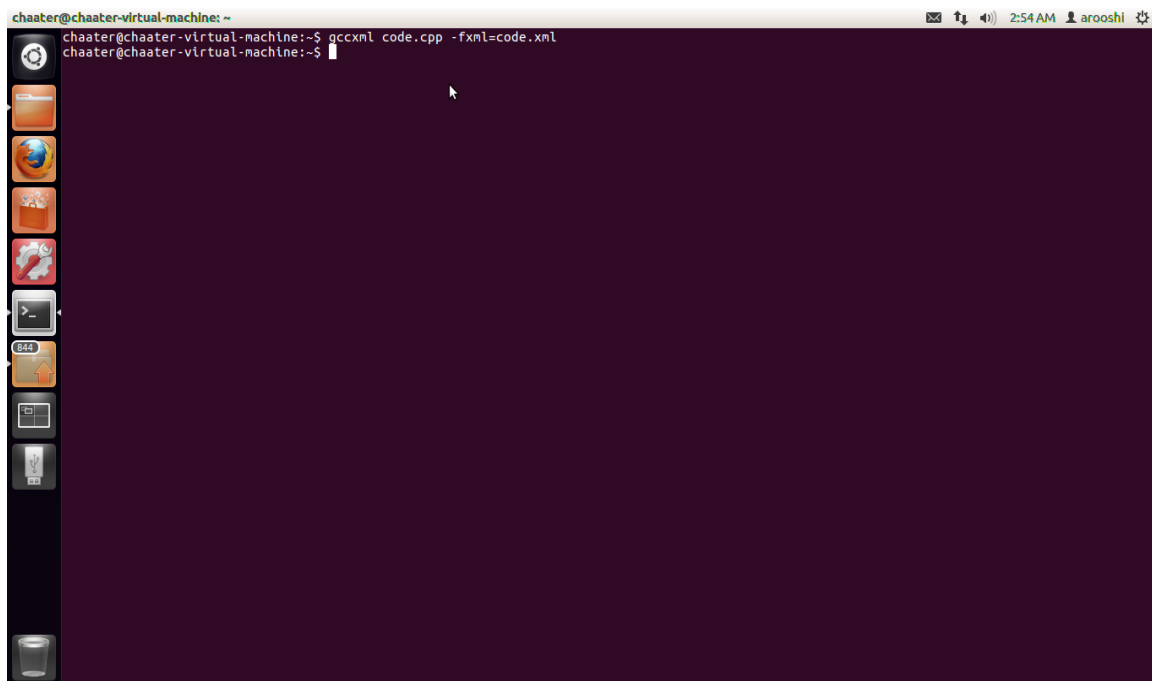


Figure 4.3 Process for running GCC XML

Figure5.3 shows that how to run the GCC XML tool The GCC XML tool has been run on the ubuntu OS. To convert the code into XML on the Terminal this command need to give “GCC XML <file _name>.cpp –fxml<file_name>.xml”.Here the file name is code.

“GCC XML code.cpp –fxml code.xml” .This proposed work the name of the banking system code is Code.cpp, with this name on terminal enter the command to convert the CPP code to XML view file .The XML generated file will be generated in the home in ubuntu OS. This is the given path to save the file in home else path can be differ through cmake in order to save the generated file. Cmake is need to run the GCC XML tool, it provide the interface to the GCC XML tool.

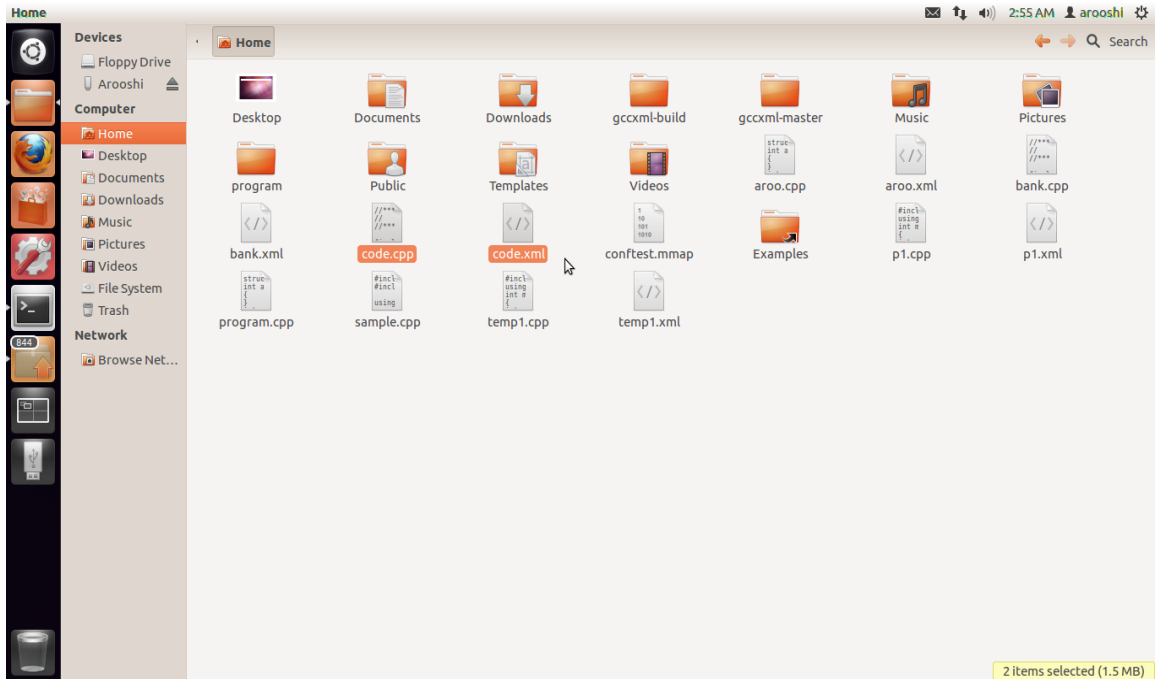


Figure 4.4 Output XML file from GCC XML

After entering that command on the terminal in ubuntu Home the XML file will be generate. Here the file name is code, so the XML generated with the name of code.

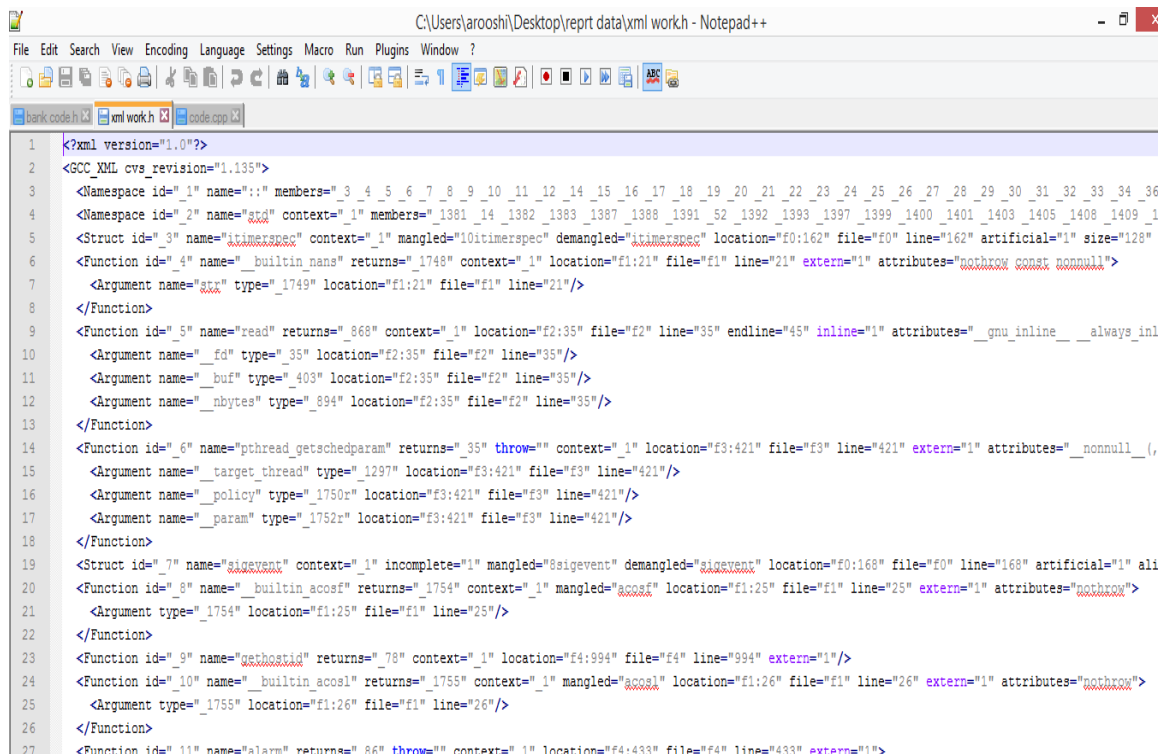


Figure 4.5 XML view file

Figure 5.5 shows the generated XML file from CPP code, above XML file does include all the data of banking system and it does also show the files include in the header file. The XML view file tells about the location of the banking system file and at which line number it does exist. It also generates the ID for each class, method and functions.

After Generating the XML view file .The XML view file is converted to the XLS (spreadsheet) file. The figure 5.6 is the XLS file of the account class in the banking system code that contains all the functions that exist in that class that are create_account, Show_account, Modify, Dep, Draw, Report, Retacno, Retdeposit and retype.

This conversion is being done with the online available tool that is “luxonsoftware”. This tool is easily available on the internet and this tool automatically generates the XLS file and provides ease of downloading the result in few minutes.

Below figure 5.6 is the XLS file that contain the data of the class account and it shows all the functions that are include in the account class and it show that from which line that function is being call and where is end in endline. In inline column it shows the calling of the function in the Main.

| | A | B | C | D | E | F | G | H | I | J | K | L | M | O |
|----|---------|----------------|---------|---------|--------|------------------------------|-------------------------------|----------|------|------|---------|--------|-------|---|
| | id | name | returns | context | access | mangled | demangled | location | file | line | endline | inline | virtu | |
| 5 | 3_2310 | create_account | _111 | _1272 | public | ZN7account14create_accountEv | account::create_account() | f35:35 | f35 | 35 | 47 | | 190 | |
| 6 | 4_2311 | show_account | _111 | _1272 | public | ZNK7account12show_accountEv | account::show_account() const | f35:50 | f35 | 50 | 56 | | 216 | |
| 7 | 4_2311 | show_account | _111 | _1272 | public | ZNK7account12show_accountEv | account::show_account() const | f35:50 | f35 | 50 | 56 | | 246 | |
| 8 | 4_2311 | show_account | _111 | _1272 | public | ZNK7account12show_accountEv | account::show_account() const | f35:50 | f35 | 50 | 56 | | 339 | |
| 9 | 5_2312 | modify | _111 | _1272 | public | ZN7account6modifyEv | account::modify() | | f35 | 60 | 70 | | 248 | |
| 10 | 6_2313 | dep | _111 | _1272 | public | ZN7account3depEi | account::dep(int) | f35:74 | f35 | 74 | 76 | | 345 | |
| 11 | 7_2314 | draw | _111 | _1272 | public | ZN7account4drawEi | account::draw(int) | f35:79 | f35 | 79 | 81 | | 356 | |
| 12 | 8_2315 | report | _111 | _1272 | public | ZNK7account6reportEv | account::report() const | f35:84 | f35 | 84 | 86 | | 313 | |
| 13 | 9_2316 | retacno | _35 | _1272 | public | ZNK7account7retacnoEv | account::retacno() const | f35:90 | f35 | 90 | 92 | | 244 | |
| 14 | 9_2316 | retacno | _35 | _1272 | public | ZNK7account7retacnoEv | account::retacno() const | f35:90 | f35 | 90 | 92 | | 214 | |
| 15 | 9_2316 | retacno | _35 | _1272 | public | ZNK7account7retacnoEv | account::retacno() const | f35:90 | f35 | 90 | 92 | | 281 | |
| 16 | 9_2316 | retacno | _35 | _1272 | public | ZNK7account7retacnoEv | account::retacno() const | f35:90 | f35 | 90 | 92 | | 337 | |
| 17 | 10_2317 | retdeposit | _35 | _1272 | public | ZNK7account10retdepositEv | account::retdeposit() const | f35:95 | f35 | 95 | 97 | | 352 | |
| 18 | 11_2318 | retype | _2109 | _1272 | public | ZNK7account7retypeEv | account::retype() const | f35:100 | f35 | 100 | 102 | | 353 | |

Figure 4.6 XLS Account class and its functions

The figure 5.7 shows that data of Main and the functions includes in it. It shows the file (F35) that in the XML where the bank file does exists. It also shows the function_id, location, line starting and endline where the function get terminate.

| | B | C | D | E | F | G | H | I | L | O | Q |
|------|-------------|-----------|------------------|---------|---------|----------|------|------|---------|-----------------------|----------------------------|
| 1 | Function_id | id | name | returns | context | location | file | line | endline | mangled | demangled |
| 194 | | 192_258 | main | _35 | _1 | f35:122 | f35 | 122 | 0 | | |
| 385 | | 383_509 | intro | _111 | _1 | f35:377 | f35 | 377 | 384 | Z5introv | intro() |
| 445 | | 443_584 | modify_account | _111 | _1 | f35:231 | f35 | 231 | 259 | Z14modify_accounti | modify_account(int) |
| 505 | | 503_665 | deposit_withdraw | _111 | _1 | f35:323 | f35 | 323 | 368 | Z16deposit_withdrawii | deposit_withdraw(int, int) |
| 663 | | 661_867 | display_sp | _111 | _1 | f35:200 | f35 | 200 | 223 | Z10display_spi | display_sp(int) |
| 875 | | 873_1140 | delete_account | _111 | _1 | f35:267 | f35 | 267 | 291 | Z14delete_accounti | delete_account(int) |
| 980 | | 978_1285 | write_account | _111 | _1 | f35:186 | f35 | 186 | 193 | Z13write_accountv | write_account() |
| 1045 | | 1043_1364 | display_all | _111 | _1 | f35:298 | f35 | 298 | 316 | Z11display_allv | display_all() |
| 1172 | | | | | | | | | | | |

Figure 4.7 XLS Main function

| | A | B | C | D | E | F | G | H | I | L | M |
|-----|-------|---------|-------|--------|---------|---------|----------|------|------|-----------|---|
| 1 | id | name | type | offset | context | access | location | file | line | GCC_XML_I | |
| 184 | _2302 | acno | _35 | 0 | _1272 | private | f35:19 | f35 | 19 | 0 | |
| 185 | _2303 | name | _4555 | 32 | _1272 | private | f35:20 | f35 | 20 | 0 | |
| 186 | _2304 | deposit | _35 | 448 | _1272 | private | f35:21 | f35 | 21 | 0 | |
| 187 | _2305 | type | _2109 | 480 | _1272 | private | f35:22 | f35 | 22 | 0 | |
| 381 | | | | | | | | | | | |

Figure 4.8 XLS files number of variable in class

In figure 5.8 the XLS file show the variable used in the account class. Those are acno, name, deposit, and type. The XLS file automatically segregates all this data in separate file in a spread sheet .According to the file number of the banking system that has been generated in XML file which is here is F35,select and find the F35 in the XLS and segregate the data of banking system file that is F35.

| | id | name | GCC_XML_Id | D |
|----|-----|--|------------|---|
| 27 | f25 | /usr/include/i386-linux-gnu/sys/types.h | 0 | |
| 28 | f26 | /usr/include/c++/4.6/new | 0 | |
| 29 | f27 | /usr/include/i386-linux-gnu/bits/sigset.h | 0 | |
| 30 | f28 | /usr/include/c++/4.6/iosfwd | 0 | |
| 31 | f29 | /usr/include/c++/4.6/bits/locale_classes.h | 0 | |
| 32 | f30 | /usr/include/_G_config.h | 0 | |
| 33 | f31 | /usr/include/i386-linux-gnu/sys/select.h | 0 | |
| 34 | f32 | /usr/include/i386-linux-gnu/bits/waitstatus.h | 0 | |
| 35 | f33 | /usr/include/i386-linux-gnu/bits/confname.h | 0 | |
| 36 | f34 | /usr/include/i386-linux-gnu/bits/select2.h | 0 | |
| 37 | f35 | bank.cpp | 0 | |
| 38 | f36 | /usr/include/locale.h | 0 | |
| 39 | f37 | /usr/include/xlocale.h | 0 | |
| 40 | f38 | /usr/include/c++/4.6/bits/cxxabi_forced.h | 0 | |
| 41 | f39 | /usr/include/i386-linux-gnu/sys/sysmacros.h | 0 | |
| 42 | f40 | /usr/include/i386-linux-gnu/bits/sched.h | 0 | |
| 43 | f41 | /usr/include/getopt.h | 0 | |
| 44 | f42 | /usr/include/i386-linux-gnu/bits/sys_errlist.h | 0 | |
| 45 | f43 | /usr/include/i386-linux-gnu/bits/setjmp.h | 0 | |

Figure 4.9 XLS file include all the header files.

Figure 5.9 show the headers files include in the code. The entire files which are available in header file has also been extracted in XML view file and when the XML file is converted to XLS file it shows all the header files includes in the code.

Once the XLS file is generated then manually according to the ID generated of banking system in XML file need to sort all the data of the banking system including classes, functions, files, variables. As above the snapshot are of the sorted data, all the above data is been manually sorted in reference to the ID generated in the XML view file. From the sorted data generated the dependency graph. The dependency graph is generated by understanding the dependency of functions in the Account class and the Function include in the MAIN. The figure 5.6 and 5.7 are the images of the account class that shows the functions included in it and main and all the functions included in it.

Figure 5.6 show the inline column which include the numeric value in front of each function .For example first row in figure 5.6 is create_account that is at the location F35:35, file 35 it and it start from line 35 and end at 47 in endline. With this there is inline column which show the inline that is 190.This 190 is the line number which represent that where it is been calling in MAIN function that is in figure 5.7 there is the function write_account that is at the location F35:35, file F35 start at 186 and end in endline 193.Which means the function in account class that is create_account located at line 190 is been calling in write_account that lie between 186-193 and 190 does lie between them, and in this way able to see the calling and dependency between classes and the functions. On the basis of this manually sorted dependency and relationship between the classes and the functions generated the dependency graph.

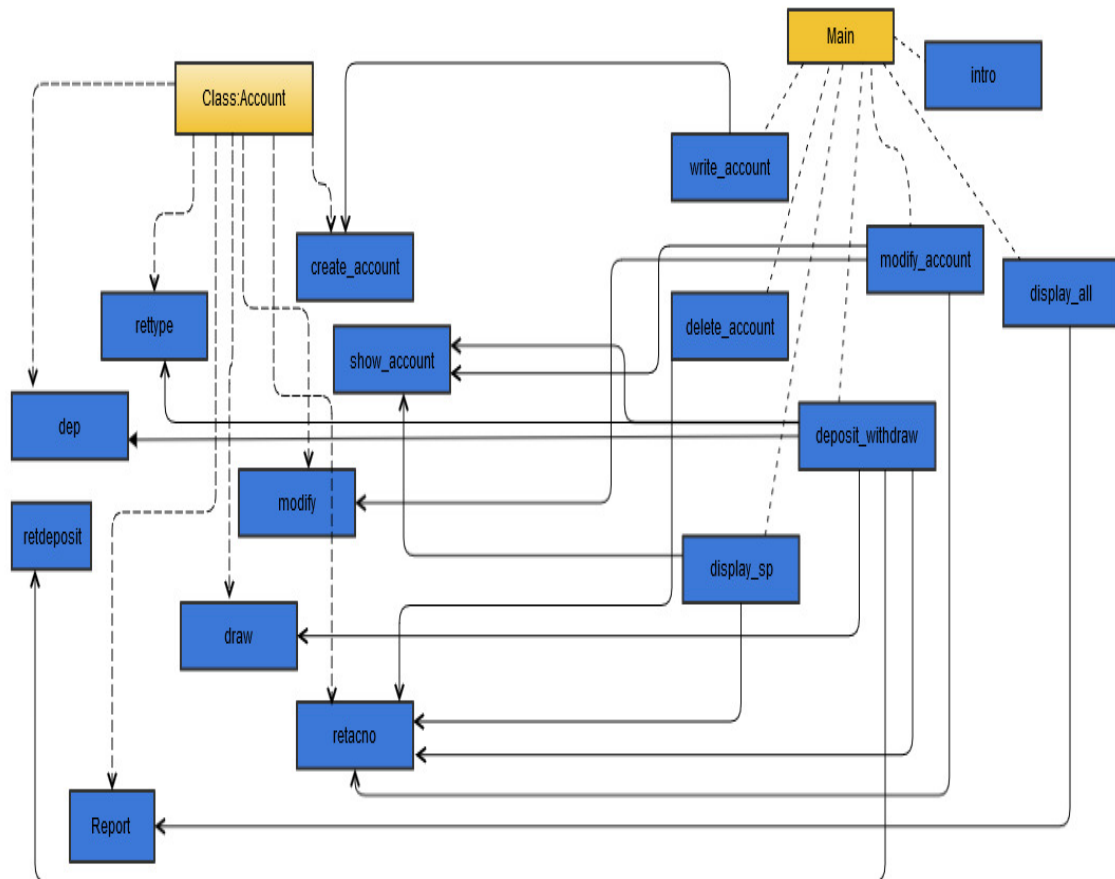


Figure 4.10 Dependency graph generated from XLS file.

Figure5.10 shows the dependency between the class account and the main function and it shows that which class is calling which function and it shows that how main function is calling the function of the account class. Dependency graph are drawn to show the

dependencies between the several objects among each other. Dependencies graph generally draw as directed graph .It help to visualize the flow of the data, calling of the several modules with each other .Dependencies graph help to represent that how the one module of the system depend on other module, how much object are interrelated and inter-dependent. It shows all the dependency between the function, calling of the functions. It provides an ease of understanding the huge line of code without the inspection of the code. It reduces the human effort to comprehend the code and it delay the generation of the error. This dependency graph help to make emendation in the code by looking at the dependency graph one can understand that by making any change in the any class or the function which other class or the functions will be effected .With the help of the dependency graph if a maintainer or the developer want to developed the new system form the existing one then instead of inspection of the code the developer or the maintainer can see the dependency graph and can understand the whole system code and can visualize that where it will be feasible to make changes and where it will affect the other modules ,classes and the function It can also be visualize that by making change in any class or the function how much other classes will be affected ,it also tell that by making change how much the system structure will be changed positively and negatively in both the ways.

This work withdraw the documentation with high understanding of the whole system and one can easily do emendations if needed without the risk of generating the mistakes or errors or affecting the functionality of other modules. This enhance the software maintenance of the system which is most difficult stage in the software cycle, it provide the understanding how a program is implemented and how its functions is a major factor when maintain the computer system. This will also generate the automated documentation which will be understood either by expertise, novice or student.

CONCLUSION AND FUTURE SCOPE

A well defined documentation with the ease of understanding is the conclusion of my proposed work. The program code of c++ will be sorted according to their interrelationship, similarities and inter and intra dependencies. The huge line of code is been understand without the inspection of the code just with the help of the dependency graph one can understand the system easily. All the dependency between the classes, function and the other modules can be seen easily.

The XML file is generated from the GCC XML tool which give the output in the form XML description of c++ code it give the output in the form of tree. The generated XML file then converted to the XLS file with the help of online tool which give the whole data separately in spreadsheet which manually been sorted according to the file id that is been generated in the XML view file. Sorting the data according to the c++ code id that is generated in XML file, give the whole data of the system separately in column in XLS file. From that entire data the dependency between the class and the main function can be visualize .With the reference of the inline and the endline it can be concluded and sorted that which class and the function is been called by which other class and the function .In this proposed work it been visualize that main function is been calling the function from the account class. It also shows the multiple calling of the function through main function. Other file also been sorted that are number of variables in the code, number of classes in the code, number of header file in the entire code, total number of functions in the code. It also show separately the number of functions in the particular class their file location and from which line number that particular function is started and ending that can be reflect in the cpp code the particular line number where the particular function does exists. It tells the exact location of the function where its been located. From the sorted XLS view file the dependency between the class and the main function is concluded manually with the reference of the inline.

The calling of each function from the class into the main function is thoroughly transparent. With the sorted data of the of the class and the main function the dependency graph is been generated manually .The dependency graph give the clear and the whole understanding of the code .It shows clearly that which class is been called by which function, it also show that how many time the function is been called by the other function or the other classes. The dependency graph shows all the dependency between all the classes, functions and the other modules.

The comprehension of the code can be done with the help of using data mining technique as well. By applying the data mining on the XLS file that is been generated the code can be comprehend in more enhanced way, instead of generating the dependency graph by using any data mining technique like K-mean, clustering the more functionality and dependencies between the module of the code can be comprehend. This proposed work can also be enhanced by generating the automated tool. This proposed work can be done by generating the automated tool, to generate the dependency graph. Because dependency graph shows the dependency among all the functions, classes variables used in that particular program. It is easy to understand or easy to developed the new system from the dependency graph of existing system.

Chapter 6

LIST OF REFERENCES

- [1] <http://www.computer.org/portal/web/swebok/html/ch6#BREAKDOWN>.
- [2] Christos Tjortjis, Loukas Sinos, Paul Layzell (2003), "Facilitating Program Comprehension by Mining Association Rules from source Code", IEEE international workshop on Program Comprehension (IWPC'03) 1092-8138/03, 2003 IEEE.
- [3] Michael L. Collard, Huzefa H. Kagdi, Jonathan I. Maletic, "An XML-Based Lightweight C++ Fact Extractor", Department of computer science Kent state University, Kent Ohio 44242.
- [4] Yiannis Kanellopoulos and Christos Tjortjis, "Data mining Source Code to Facilitate Program Comprehension: Experiments on Clustering Data Retrieved from c++ Programs", Department of Computation, UMIST, Manchester, UK.
- [6] A. Von Mayrhauser and A.M. Vans, Program Understanding – A Survey, Technical Report CS-94-120, Dept. of Computer Science, Colorado State University, August 1994.
- [7] T.M. Pigoski, Practical Software Maintenance: Best Practices for Managing your Software Investment, Wiley Computer Publishing, 1996.
- [8] K. Sartipi, K. Kontogiannis and F. Mavaddat, 'Architectural Design Recovery Using Data Mining Techniques', Proc. 2nd European Working Conf. Software Maintenance Reengineering (CSMR 2000), IEEE Comp. Soc. Press, 2000, pp. 129-140.
- [9] I. Sommerville, Software Engineering, 6th edition, Harlow, Addison-Wesley, 2001.
- [10] F. Tip, "A Survey of Program Slicing Techniques", Technical Report CS-R9438, Centrum voor Wiskunde en Informatica, Amsterdam, 1994.
- [11] C. Tjortjis, N. Gold, P.J. Layzell and K. Bennett, "From System Comprehension to Program Comprehension", Proc. IEEE 26th Int'l Computer Software Applications

Conf. (COMPSAC 02), IEEE Comp. Soc. Press, 2002, pp.427-432.

- [12] U. Dekel, “Applications of Concept Lattices to Code Inspection and Review”, 1993.
- [13] S. Kuznetsov and S. Ob’edkov, “Comparing Performance of Algorithms for Generating Concept Lattices”, ICCS’01 Int’l. Workshop on Concept Lattices–based KDD, 2001.
- [14] A. Marcus, V. Rajlich, J. Buchta, M. Petrenko, and A. Sergeyev, “Static Techniques for Concept Location in Object-Oriented Code”, 2005.
- [15] D.C.C.POO, National University of Singapore, Chee Seong Tan, “Learn To Program Java”, Second Edition, 2005.
- [16] H.P.Phyu, T.T.S.Nyunt “Concept-based Source Code Analysis for Software Maintenance”, in Proceedings of the 11th International Conference on Computer Applications (iCCA, 2013), Yangon, Myanmar, February 2013, pp. 339-343.
- [17] G. Snelling, F. Tip, “Reengineering Class Hierarchies Using Concept Analysis”, ACM, 1998.
- [18] T.Tilley, R. Cole, P. Becker, “A Survey of Formal Concept Analysis Support for Software Engineering Activities”, 2005.
- [19] S. Wang, Z. Chen, D. Wang, “An Algorithm based on Concept-Matrix for Building Concept Lattice with Hasse”, IEEE , 2007.
- [20] R. Wille, “Formal Concept Analysis as Mathematical Theory of Concepts and Concept Hierarchies”, Springer-Verlag Berlin Heidelberg, pp. 1– 33, 2005.
- [21] R. Baecker, A. Marcus, Human Factors and Typography for More Readable Programs, ACM Press, Addison-Wesley Publishing Company, Reading, MA, 1990.
- [22] R.M. Baecker, Sorting out sorting (16mm _lm), ACM SIGGRAPH '81, 1981.
- [23] B. Bederson, J. Hollan, Pad++: A zooming graphical interface for exploring alternate interface physics, in Proceedings of ACM UIST'94, Marina del Rey, California, November 1994, pp. 17{26.
- [24] M.S.K. Brade, M. Guzdial, E. Soloway, Whorf: A visualization tool for software maintenance, in

Proceedings 1992 IEEE Workshop on Visual Languages, Seattle, Washington, September 15-18, 1992, pp. 148-154.

[25] M. Brady, *The Monopoly Book: Strategy and Tactics of the World's Most Popular Game*, David McKay Company, Inc., New York, 1974.

[26] R. Brooks, Towards a theory of the comprehension of computer programs, *Int. J. Man-Mach. Stud.* 18 (1983) 543-554.

[27] M.H. Brown, Exploring algorithms using Balsa-II, *Computer* (May 1988) 136-157.

[28] M.H. Brown, ZEUS: a system for algorithm animation and multi-view editing, in *Proceedings of the IEEE 1991 Workshop on Visual Languages*, Kobe, Japan, October 1991, pp. 4-9.

[29] M.H. Brown, M.A. Najork, Algorithm animation using 3d interactive graphics, in *UIST, Proceedings of the ACM Symposium on User Interface Software and Technology*, November 1993, pp. 93-100.

[30] J. Cross, S.M. and T.D. Hendrix, The control structure diagram: an initial evaluation, *Empirical Software Eng.* 3(2) (1998) 131-156.

[31] S.M. Dray, Practical observation skills for understanding users and their work in context, in *Presented at CHI (Computer Human Interaction) 1999*. May 1999.

[32] G. Furnas, Generalized eye views, in *Proceedings of ACM CHI'86*, Boston, MA, April 1986, pp. 16-23.

Appendix

XML: Extensible Markup Language

CPP: C plus plus

GCC: GNU Compiler Collection

XLS: Excel spreadsheet

Retacno: return account number

Retdeposit: return balance amount

Rettype: return type of account

Intro: introductory screen function