AN OVERVIEW REPORT ON SOFTWARE RE- AND REVERSE ENGINEERING

Srinivasa Reddy A.[1] Murali Krishna A.[2] Srinivasulu P.[3]

[1]Asst Prof in IT, MLEC Singarayakonda, India
[2]Professor in CSE, MLEC Singarayakonda, India
[3]Assoc Prof in CSE, MLEC Singarayakonda, India
Email: [1]srinivas.asr@gmail.com

Abstract

Instability is the nature of any application. An Application may strive for and helps a business or company for some time (may be 10 or 15 years), during that time it has been corrected, adapted and enhanced many times. But every time a change is made into the application, unexpected and serious side effects occur. Yet the application must continue to evolve. Unmaintainable software is not a new problem. In the following sections, we are going to analyze the Software Re-engineering and Reverse-engineering processes and how they help any application to maintain the Quality standards.

Keywords: Instability, Re-engineering, Reverse-engineering, Software, Quality.

I. INTRODUCTION

Any Application tends to change in time according to the Business needs and Customer Requirements. If we are able to meet the customer satisfaction levels, then we can say that our product or application is Quality one. In simple terms we can define the quality as Customer Satisfaction (1). Many attributes may contribute to quality like compliant product, performance, time, Lines of code, less errors, budget etc. To achieve all the way the quality of an application or product we have to enhance the existing application to the customer level and Software Re- and Reverse engineering processes (2) helps us to achieve this. Software re-engineering is concerned with re-implementing legacy systems to make them more maintainable. Re-engineering may involve re-documenting the system, organizing and restructuring the system, translating the system to a more modern programming language and modifying and updating the structure and values of the system's data. The functionality of the software is not changed and, normally, the system architecture also remains the same.

The term *software engineering* first appeared in the 1968 NATO Software Engineering Conference and was meant to provoke thought regarding the current "software crisis" at the time. Software is set of programs and rules associated with particular product and Software Engineering as the application of a systematic, disciplined, quantifiable approach to the development, operation, and maintenance of software, and the study of these approaches; that is, the application of engineering to software.

II. REENGINEERING

Software re-engineering (3) is concerned with re-implementing legacy systems to make them more maintainable. Re-engineering may involve re-documenting the system, organizing and restructuring the system, translating the system to a more modern programming language and modifying and updating the structure and values of the system's data. The functionality of the software is not changed and, normally, the system architecture also remains the same. Most of the Reengineering works depends on the data processing. Data Mining and Ware housing helps us to find out the information which is processes data and which tells the customer behavior and customer needs which is very important in improving the Quality of the product.

From a technical perspective, software re-engineering may appear to be a second-class solution to the problems of system evolution. The software architecture is not updated so distributing centralized systems is difficult. It is not usually possible to radically change the system programming language so old systems cannot be converted to object-oriented programming languages such as Java or C++. Inherent limitations in the system are maintained because the software functionality is unchanged.

However, from a business point of view, software re-engineering may be the only viable way to ensure that legacy systems can continue in service. It may be

too expensive and too risky to adopt any other approach to system evolution. To understand the reasons for this, we must make a rough assessment of the legacy system problem.

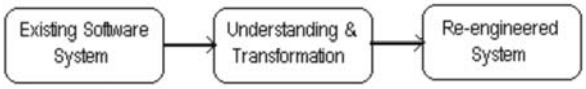

Fig 1: Software Re-engineering

Re-engineering a software system has two key advantages over more radical approaches to system evolution:

1. Reduced risk: T here is a high risk in re-developing software that is essential for an organization. Errors may be made in the system specification; there may be development problems, etc.

2. Reduced cost: T he cost of re-engineering is significantly less than the costs of developing new software.

The term re-engineering is also associated with business process reengineering (BPR). Business process re-engineering is concerned with redesigning business processes to reduce the number of redundant activities and improve process efficiency. It is usually reliant on the introduction or the enhancement of computer-based support for the process. Process re-engineering is often a driver for software evolution as legacy systems may incorporate implicit dependencies on the existing processes.

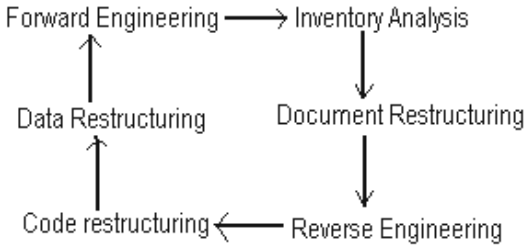

Fig 2: Software Re-engineering Process Model

Inventory Analysis provides all the information that provides a detailed description (size, age, criticality, budget, personal etc.) of Application.

Document Restructuring mainly deals with problems of weak documentation which is very important for any business organization like Software Requirement Specification (SRS) Document.

Reverse Engineering derives one or more design and manufacturing specifications for a product by examining actual specimens of the product.

Code Restructuring mainly deals with legacy systems which have solid program architecture, but individual modules were coded in a way that makes them difficult to understand, test and maintain.

Data Restructuring deals with a program effect of weak data architecture which is very difficult and enhance.

Forward Engineering deals with applications which can be rebuilt using an automated "reengineering engine".

Forward engineering starts with a system specification and involves the design and implementation of a new system.

Re-engineering starts with an existing system and the development process for the replacement is based on understanding and transformation of the original system.

Fig 3: Forward Engineering

Software Re-engineering mainly deals with Software Maintenance which includes the four activities: Error Correction, Adaptation, Enhancement and Re-engineering. Only about 20 percent of all maintenance work is spent on "fixing mistakes". The remaining 80 percent is spent adapting existing systems to changes in their external environment, making enhancements requested by users and reengineering an application for future use. This is known as 20-80 rules in software engineering.

III. REVERSE ENGINEERING

Reverse engineering (4) is the process of analyzing software with the objective of recovering its design and specification. The program itself is unchanged by the reverse engineering process. The software source code is usually available as the input to the reverse engineering process. Sometimes,

however, even this has been lost and the reverse engineering must start with the executable code.

Reverse engineering is not the same thing as re-engineering. The objective of reverse engineering is to derive the design or specification of a system from its source code. The objective of re-engineering is to produce a new, more maintainable system.

Reverse engineering is used during the software re-engineering process to recover the program design which engineers use to help them understand a program before re-organising its structure. However, reverse engineering need not always be followed by re-engineering (5).

Three Reverse Engineering issues must be addressed: Abstraction level, Completeness and Directionality. The reverse engineering process starts with an analysis phase. During this phase, the system is analysed using automated tools to discover its structure. In itself, this is not enough to re-create the system design.

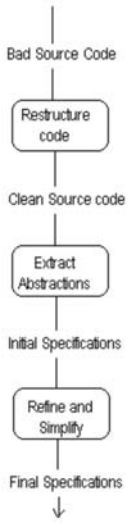

Fig 4: Reverse Engineering

Engineers then work with the system source code and its structural model. They add information to this which they have collected by understanding the system. This information is maintained as a directed graph that is linked to the program source code.

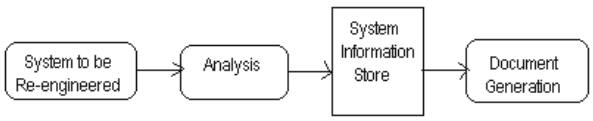

Fig 5: Reverse Engineering Process

In implementing Reverse engineering, we should focus on Internal Data Structure and Database Structure. Reverse engineering helps us to understand processing (6).

Reverse engineering (RE) is the process of discovering the technological principles of a device, object or system through analysis of its structure, function and operation. It often involves taking something (e.g., a mechanical device, electronic component, or software program) apart and analyzing its workings in detail to be used in maintenance or to try to make a new device or program that does the same thing without utilizing any physical part of the original.

Reasons for reverse engineering:

1. Interoperability
2. Lost documentation
3. Product analysis
4. Digital update/correction
5. Academic/learning purposes
6. Curiosity
7. Learning
8. Lost Code
9. Understandability

IV. SAMPLE EXAMPLE

Let us understand the importance of the Re- and Reverse engineering concepts using sample example from Basic Language C.

Let us Take Example of addition program.

Program

```
#include<stdio.h>
#include<conio.h>
Void main()
{
int salary,basic,da;
clrscr();
printf("Enter basic,da: ");
scanf(""%d%d",&basic,&da);
salary=basic+da;
printf("salary=%d",salary);
getch();
}
```

Output

Enter basic,da: 8000 5000

Sum=13000

This is the Output of addition program. Here we have to test the program using all possible test cases.

When we are trying to add two numbers, it's important to consider the range of values of individual variables.

In the example, all salary, basic and da has the values of – 32768 to +32767. Here in the program, if the salary don't exceeds 32767, This program will work fine, what happens if salary exceeds 32767.

So we have to change this program or re-engineer the program to store more values:

Unsigned int *a*, *b*, *c*;

Changing the program depends on the time and circumstances of the customer and here the variable salary.

What is the meaning of Reverse engineering in this case?

When ever we compile a program .BAK and .EXE files are created along with .C program. In case of lost code or lost documentation of related to code or lost of .BAK and .C files. We can understand the program thru .EXE files. We can observe the display statements and we can find out the statements in the code. Analyzing the output also informs the programmer about break points and necessary action to improve the program quality.

Hence we can say that Re-Engineering along with Reverse engineering helps a programmer or Software Engineer or Developer to make quality product which will be satisfied by different customers according to time and place.

V. CONCLUSION

A software engineer performing a reengineering activity must typically understand and manage three forms of information:

- The structure of the existing source,
- The structure of the desired reengineering source, and
- The relationship between the reengineered and existing structures.

Existing source code analysis and reverse engineering tools do not provide adequate support to the engineer in all of these dimensions.

Whereas traditional software engineering primarily focuses on "doing it right the first time," reverse engineering addresses the expensive area of maintenance, where one pays the cost of not having done it right the first time, or allowing it to decay over time. Software reverse engineering, or

Program comprehension is the difficult task of recovering design and other information from a software system. It is difficult to perform because there are intrinsic difficulties in performing the mapping between the language of high level design requirements and the details of low level implementation. Although reverse engineering depends heavily on the human in the loop, there are a variety of automation methodologies available for support.

REFERENCES

[1] B.W. Boehm, *Software Engineering Economics*, Prentice-Hall, Englewood Cliffs, NH, 1981.

[2] E.J. Chikofsky and J. H. Cross, II, "Reverse Engineering and Design Recovery: A Taxonomy," *IEEE Software*, vol. 7, no. 1, pp. 13-17, January 1990.

[3] M. Hammer and J. Champy, "Reengineering the Corporation: A Manifesto for Business Revolution," Harper- Business, New York, 1993.

[4] K. Kamper and S. Rugaber, "A Reverse Engineering Methodology for Data Processing Applications," Georgia Tech Technical Report GIT-SERC-90/02, March 1990.

[5] Software Engineering, A practitioner's Approach- Roger S. Pressman, 6th edition.McGrawHill International Edition.

[6] Software Engineering- Sommer ville, 7th edition, Pearson education.

Mr. A.Srinivas Reddy received B.Tech from JNTU, Hyderabad and M.Tech from Sathyabama University, Chennai. He has published more than 15 papers in both international and National Conferences/Journals. His research interest includes Software Engineering, Mobile Ad-Hoc Networks and Data Mining.