Reverse Engineering

Aabhas Singhal
Department of CSE
Nirma University – Institute Of Technology
Ahmedabad, India

Shlok Gandhi
Department of CSE
Nirma University – Institute Of Technology
Ahmedabad, India

ABSTRACT
This paper provides the basic information about Reverse Software Engineering and its advantages and disadvantages of Reverse Engineering. Today Reverse Engineering is used in many fields of Information Technology in form of Legacy compatibility, Malware Analysis, Network Analysis, Binary code patching, debugging, and improvising existing algorithms, rapid prototyping and even software reusability. The paper provides understanding of Reverse Engineering and discusses some of the advantages and issues in detail.

Keywords
Reverse Engineering, Reusability, Disassembly, and Decompilation.

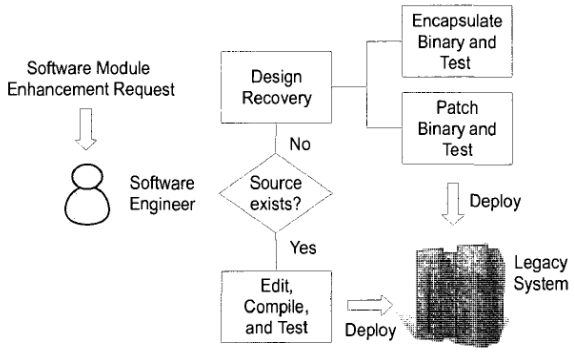
1. INTRODUCTION
Engineering is profession involves in constructing, designing, manufacturing, maintaining of systems, products and structures. Basically there are two types of Engineering: Forward and Reverse. Reverse Engineering is a process of understanding, duplicating an al- ready existing product or component without the aid of drawings, documentation or computer model[6]. It is process of analyzing a system for three main purposes:

- Identify the components and understanding their interrelationship.
- Create their representations in another form.
- Generate physical representation of the system.

In Software Engineering, the term reverse engineering usually means going backward through the development cycle and under- standing the implementation and working of the software. Reverse engineering is a phase preceding the re-engineering. According to Chikofsky and Cross ”Reverse engineering is the process of analyzing a subject system to create representations of the system at a higher level of abstraction.[1] In this case, the o/p of implementation phase is reverse engineered to the analysis phase, an inverse strategy of traditional, basic waterfall model. Reverse engineering is process of examination only and not modifying the source code (which is a part of re-engineering).In practice, there are two types of Software Reverse engineering. One in which, source code is readily available, but higher level aspects of the programs are poorly documented or unavailable and they are to be discovered. Secondly the cases in which source code is not available and the only objective of Reverse engineering is to discover the source code

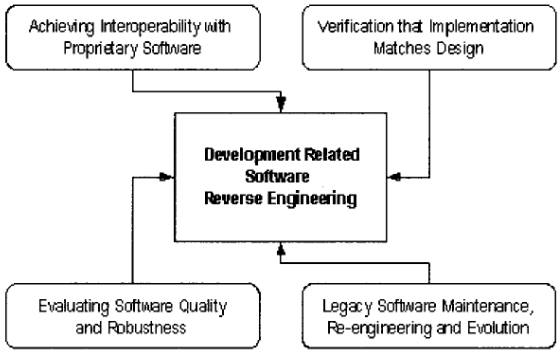
Black Box Testing in s/w engineering is somewhat related to Reverse engineering. The tester has access to the code and his goal is detect bugs and undocumented features by accessing the component from outside. Other uses of Reverse engineering are security auditing, removal of copy protection, bypassing the access permissions, customize the embedded system, in house repairs, enabling additional features and also personal customization

2. WHY REVERSE ENGINEERING?
While most of the software code that has been written today is not into use, but a considerable amount of it has survived the generation and continues to be a part of global economy. In 1997, the Gartner Group reported that 80percent of the world’s business ran on COBOL with over 200 billion lines of code in existence and with an estimated 5 billion lines of new code annually [2]. Since re-coding all this software is not a feasible or economical solution, the only reasonable option is to maintain and evolve the code, mostly with concepts of Reverse Software Engineering. While Scientists or software engineers are busy in maintaining legacy systems, more than half of their time is spent in understanding the existing code. Spending this time in understanding the program is not economically feasible as software industry grows in complexity and size. To lessen this time of software engineers, it is advisable to practice reverse engineering techniques to improve ability of understanding the program quickly and efficiently.



Though several computer aided soft wares are available for the purpose, the reverse engineering tools help transferring the information of software’ design into software developers’ mind. The expectation is that the developer would be able to implement, improvise the understood design and integrate the information to build a system model. Software tools can never beat the model of mental understanding. The main problem with software maintenance is that it cannot be misspelled with some clever technique. [3] argues ”re-engineering code to create a system that will not need to be re- verse engineered again in the future is presently unattainable.”

According to [4], there are four software development related reverse engineering aspects; aspects that cover a broad spectrum of software activities, including software maintenance, re-use, re-engineering, evolution, interoperability and testing. The image below summarizes the software development related reverse engineering aspects:



The following are tasks one might perform in each of the reversing scenarios [4]:

• Achieving Interoperability with Proprietary Software: Develop applications or device drivers that interoperate (use) proprietary libraries in operating systems or applications.

• Verification that Implementation Matches Design: Verify that code produced during the forward development process matches the envisioned design by reversing the code back into an abstract design.

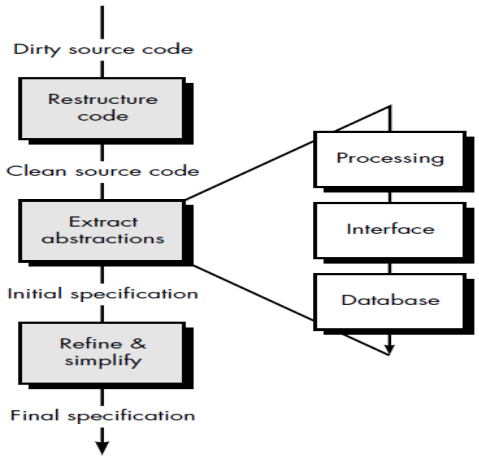
• Evaluating Software Quality and Robustness: Ensure the quality of software is- fore purchasing it by performing heuristic analysis of the binaries to check for certain instruction sequences that appear in poor quality code.

• Legacy Software Maintenance, Re- engineering, and Evolution: Recover the design of legacy software modules when source is not available to make possible the maintenance, evolution, and reuse of the modules.

3. REVERSE ENGINEERING PROCESS

The Reverse Engineering process is shown in the figure below. The unstructured code is structured such that it contains only structures programming constructs, before even commencing the Reverse Engineering process. This makes the code easier to be understood and read and provides the basis of Reverse Engineering activities.

[5]The main core activity of Reverse Engineering is extract abstractions. The designer/engineer must evaluate the primitive program and thereby structured code, and extract meaningful specification of processing that is to be performed, interface that is to be built and database that is to be used



3.1 Understand processing

The first step of Reverse Engineering is to ex- tract and understands the procedural specifications of the already existing application. To understand the procedural abstractions, the code is analyzed and reviewed at different levels of abstraction: system, component, program, statement and pattern. The all-round functionality of the entire program is to be fully under- stood by the designer/engineer. This provides the insight of the interdependence of the individual components of the application. A block diagram of interdependence of the functional abstractions is created. Each of the subcomponent performs some function and represents some procedural abstraction. If in some cases the abstractions are already present; then they are verified and modified if needed.

3.2 Understand Data

As Reverse Engineering is a process of different level of abstractions, at software level, internal data structures are often to be Reverse Engineered as a part of the process. Reverse Engineering operations for understanding internal data structures focus on definition of classes of objects. It is often done by identifying flags and internal/local data structures of the application, defining the interrelationship between the flags and the data structures (local or global) and list all variables that are interdependent in any way and scope.

Even internal database structures are important to be Reverse Engineered. In this case it almost understands the data objects and their interrelationship. It can be done by building an initial model, listing and determining its candidate keys, redefinition of tentative class, defining generalization and discovering associations.

3.3 Understanding the User Interface

Simple and sophisticated User Interfaces have become a vital part of any applications redeveloping them is a common part of the Reverse Engineering process. Engineers must fully understand the existing interface and also it's structural and behavioral aspects. We must be clear about three things before Reverse Engineering of an application:

1] Basic Options

2] Behavioral response of the application

3] Possible and better replacements

4. IS REVERSE ENGINEERING LEGAL? ETHICAL?

There are two main legal aspects with Reverse Engineering:

1] Copyright Infringement: Related to shape and look of the object

2] Patent Infringement: Related to idea of functioning of the product

According to some, patent is nothing more than a warning sign for a competitor to dis- courage competition. If the idea is useful enough, the competitor may do the following: Negotiate with the original person a license to lend the idea, Claim that the idea is no big deal and a normal obvious idea for anyone doing work in that field or make a minor change in the product and claim that its new changed product

For this Cleanroom Reverse Engineering is conducted so as to carry out sequential steps:

1] A team of trained engineers would perform disassembly and Decompilation of the soft- ware, investigate the code and describe them in as much detail as possible.

2] Description is documented properly and passed on to new group of engineers who have no knowledge of the previous existing software.

3] The second group of Engineers would develop the software based on the documentation provided by the first group so as to have different and unchallenged approach.

So this will probably avoid the law infringements and theft of idea. Some of the new courts suggests that Reverse Engineering done for achieving interoperability and betterment through an independent procedure is totally legal Is Reverse Engineering Ethical or not is a largely debated topic and does not have a clear solution. But majority believes that it is ethical as programs can be advances and that is not an intellectual property. But there are incidents where Software companies have been hurt by the Reverse Engineering process.

5. CONCLUSION

Reverse engineering makes the system structure better, creates new dimension of system documentation and makes it easier, more lucid to understand. Reverse engineering a software application program has advantages over more radical approaches for helping system better- mint and evolution. The main disadvantage of software reverse engineering is that there are practical limits to the extent that a system can be improved by reverse engineering.

6. ACKNOWLEDGMENTS

The authors would like to thank Prof. Sharing Pandya, Nirma University - Institute Of Technology for his immense support throughout the study on this subject.

7. REFERENCES

[1] Chikofsky, E. J.; Cross, J. H. (January 1990). ”Reverse engineering and design recovery: A taxonomy”. IEEE Software 7: 1317. doi:10.1109/52.43044.

[2] L. Cunningham. (2008, Jul 9). COBOL Reborn [Online]. Available: http://it.toolbox.com/blogs/oracle- guide/cobol-reborn-25896

[3] B. W Weide, W D. Heym, J. E. Hollingsworth, ”Reverse engineering of legacy code exposed,” in Proc. 17th Int. Conf. Software Engineering, Seattle, Washington, WA, 1995, pp.327-331.

[4] E. Eliam, Secrets of Reverse Engineering, Indianapolis, IN: Wiley, 2005.

[5] Software Engineering, A practitioner’s Approach by Roger S Pressman (5th Edi)

[6] Wikipedia (2014), Reverse Engineering