E-ISSN: 2278–179X

JECET; March - May 2013; Vol.2.No.2, 364-369.

Journal of Environmental Science, Computer Science and Engineering & Technology

An International Peer Review E-3 Journal of Sciences and Technology

Available online at www.jecet.org

Computer Scence

Research Article

Reverse Engineering of Software and Types Hazards

Lalita M .Lokhande* and N.V. Kalyankar

Department of Computer Science, Yeshwant Mahavidyalaya, Nanded,-431601 (MH) India

Received: 11 April 2013**; Revised:** 02 May 2013; **Accepted:** 6 May 2013

Abstract: The paper based on various hazards based on experiment of reverse engineering programs. Reverse engineering processes was used as part of a software development, an implementation of a better program and upgrade the existing documentation. The development, design information was extracted from the source codes and entered into a software development environment. The valuated design information was used to implement a new version of the software program. The experiments carried out by recovering the information and it will implemented, dealing with some incomplete part of information and reverse engineering. The reverse engineering process used to recover the design; hazards and the experience gained during the study are reported.

Keywords: Hazards, Reverse engineering, forward engineering.

INTRODUCTION

The implementation of experiment derived from a single problem or multiple problems given an existing software system. How you can improve your system and how it can be developed? The problem of reimplmenting an existing programme system in a different programming language has been around for last 20 years it is based on: a) manually rewrite the programme for software's, b) Use a different language translator, c) Redesign and reimplement the software programme[1-6].

Another reason for reverse engineering is to compress product development times. In the intensely competitive global market, manufacturers are constantly seeking new ways to shorten lead-times to market a new product. Rapid product development refers to recently developed technologies and techniques that assist manufacturers and designers in meeting the demands of reduced product development time. By using

reverse engineering, model can be quickly captured in digital form, re-modeled and exported for rapid prototyping/tooling or manufacturing.

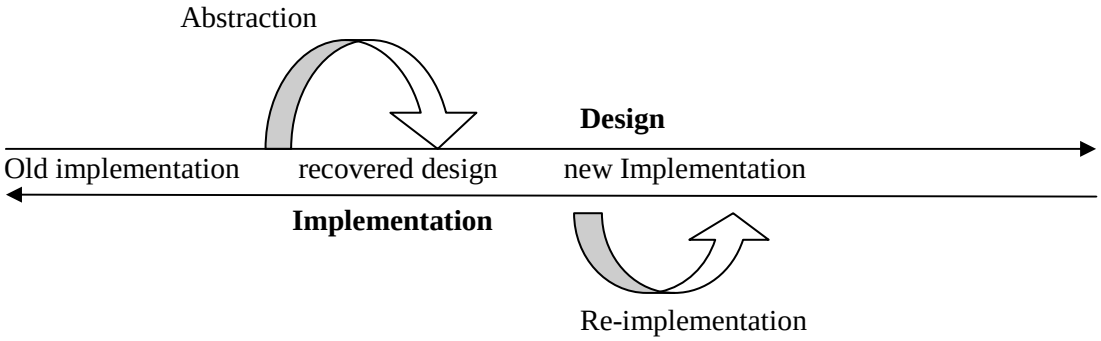

Fig.1: Software Reverses Engineering

REVERSE ENGINEERING

Reason for reverse engineering is to compress product development times. In the intensely competitive global market, manufacturers are constantly seeking new ways to shorten lead-times to market a new product. Rapid product development refers to recently developed technologies and techniques that assist manufacturers and designers in meeting the demands of reduced product development time. By using reverse engineering, model can be quickly captured in digital form, re-modeled and exported for rapid prototyping/tooling or manufacturing[7-9].

Some important reasons are part or product for reverse engineering: a) The manufacturer of an original product is a no longer produces a product. b) The original design is inadequate documentation. c) The original manufacturer no longer exists, but a customer needs the origin of product. d) The original design documentation has been lose or never existed. e) Some bad features of a product needed to be redesigned. f) To strengthen the good features of a product based on long-term usage of product. g) To analyze the good and bad features of competitors product. h) To explore new avenues to improve product performance and features. i) To gain competitive benchmarking methods to understand competitor's products and develop better products. J) The original model is not sufficient to support modifications or current manufacturing methods. k) The original supplier is unable or unwilling to provide additional parts. l) The original equipment manufacturers are either unwilling or unable to supply replacement parts or demand inflated costs for sole-source parts. m) To update obsolete materials or antiquated manufacturing processes with more current, less-expensive technologies[10-12].

Reverse engineering enables the duplication of an existing part by capturing the component's physical dimensions, features and material properties. Before attempting reverse engineering a well-planned life-cycle analysis and cost/benefit analysis should be conducted to justify the reverse engineering projects. Reverse engineering is typically cost effective only if the items to be reverse engineered reflect a high investment or will be reproduced in large quantities. Reverse engineering of a part may be attempted even if it is not cost effective, if the part is absolutely required and is mission-critical to a system. A common misperception regarding reverse engineering is that it is used for the sake of stealing or copying someone else's work. Reverse engineering is not only used to figure out how something works, but also the ways in which it does not work[13, 14].

***Examples of the different uses of reverse engineering include: (*a)** Understanding how a product works more comprehensively than by merely observing it. b) Investigating and correcting errors and limitations in existing programs. c) Studying the design principles of a product as part of an education in engineering. d) Making products and systems compatible so they can work together or share data. e) Evaluating one's own product to understand its limitations. f) Determining whether someone else has literally copied elements of one's own technology. g) Creating documentation for the operation of a product whose

manufacturer is unresponsive to customer service requests. h) Transforming obsolete products into useful ones by adapting them to new systems and platforms. In the context of software engineering, the term reverse engineering was defined and the process of analyzing a subject system to (i) identify the system's components and their inter-relationships and (ii) create representations of the system in another form or at a higher level of abstraction. Thus, the core of reverse engineering consists in deriving information from the available software artifacts and translating it into abstract representations more easily understandable by humans.[15,16]

Examples of problem areas where reverse engineering of software has been successfully applied they are: (a)Re-documenting programs and relational databases. b) Identifying reusable assets. c) Recovering architectures of programmer. d) Recovering design patterns of data. e) Building traceability between data code and documentation's. f) Identifying clones. g) Code smells and aspects of data. h) Computing change impacts. i) Reverse engineering primary and binary codes. j) Renewing user interfaces. k) Translating a program from one language to different language. l) Migrating or wrapping legacy code of software. Although software reverse engineering originated in software maintenance, its definition is sufficiently broad so as to be applicable to many problem areas, for example to create representations necessary for testing purposes or to audit security and vulnerability.

The practice of reverse engineering is widespread throughout the software industry. However, reverse engineering is also associated with hackers and crackers. The various computer law breakers as a means to subvert security steal secrets and destroy data. Software companies fear (and rightly so) that their trade secret algorithms. The methods will be more directly revealed through reverse engineering than they are through external machine observation. However, there is no general-purpose law against reverse engineering. Because reverse engineering is a crucial step in removing copy protection schemes, there is some confusion regarding its legality. The objective of the process of software reverses engineering to improve the maintainability of a software system. When you have read this chapter, you will: i) understand why re-engineering is sometimes a cost-effective option for software system evolution, ii) understand the activities such as reverse engineering and program restructuring which may be involved in the software reverse engineering process and iii) understand the differences between software and data reverse engineering and understand why data reverse engineering is an expensive and time consuming process.[17,18]

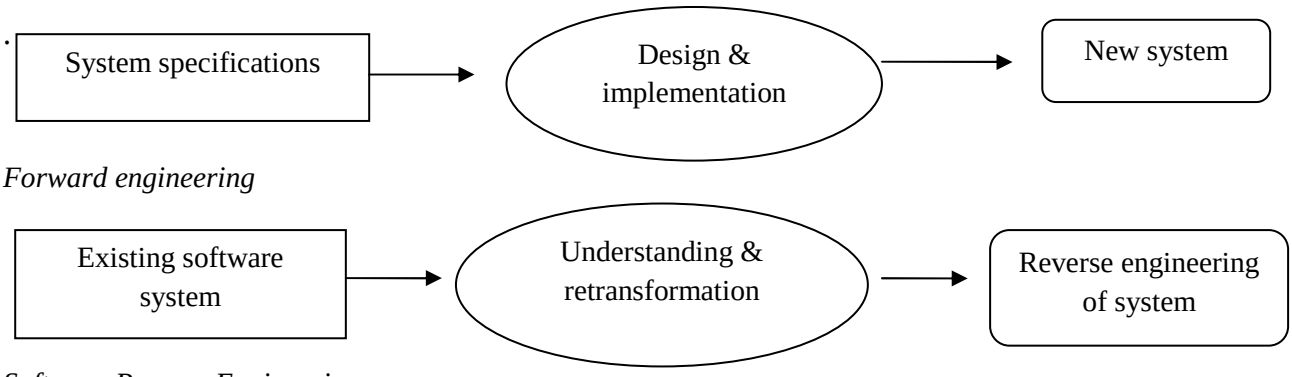

Fig. 2: Implementation of forward and software reverse engineering

Software systems which are essential for business process support. Companies rely on these systems so they must keep them in operation. Software evolution strategies include maintenance, replacement and architectural evolution of software reverse engineering. Software reverse engineering is concerned with re-implementing legacy systems to make them more maintainable. Reverse 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. From a technical perspective, software reverse engineering may appear to be a second-class solution to the

problems of system evolution. There are so many systems in existence that complete replacement or radical restructuring is financially unthinkable for most organizations. Reverse engineering a system is cost effective when it has a high business value but is expensive to maintain. Reverse engineering improves the system structure, creates new system documentation and makes it easier to understand. Re-engineering a software system has two key advantages over more radical approaches to system evolution: *Reduced risk* there 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. for example: *Reduced cost:* The cost of reverse engineering is significantly less than the costs of developing new software. A commercial system where the re-implementation costs were estimated at one lakh rupees. The system was successfully reverse engineered for twenty thousand rupees only. If these figures are typical, it is about 4 times cheaper to reverse engineer than to re-write[19].

Fig.3.The input to the process is a legacy program and the output is a structured, modularized version of the same program. At the same time as program re-engineering, the data for the system may also be reverse engineered. The activities in this reverse engineering process are: a) *Source code translation:* The program is converted from an old programming language to a more modern version of the same language or to a different language. b) *Reverse engineering:* The program is analyzed and information extracted from it which helps to document its organization and functionality. c) *Program structure improvement:* The control structure of the program is analyzed and modified to make it easier to read and understand. d) *Program modularization:* Related parts of the program are grouped together and, where appropriate, redundancy is removed. In some cases, this stage may involve architectural transformation as discussed. e) *Data re-engineering:* The data processed by the program is changed to reflect Program changes.

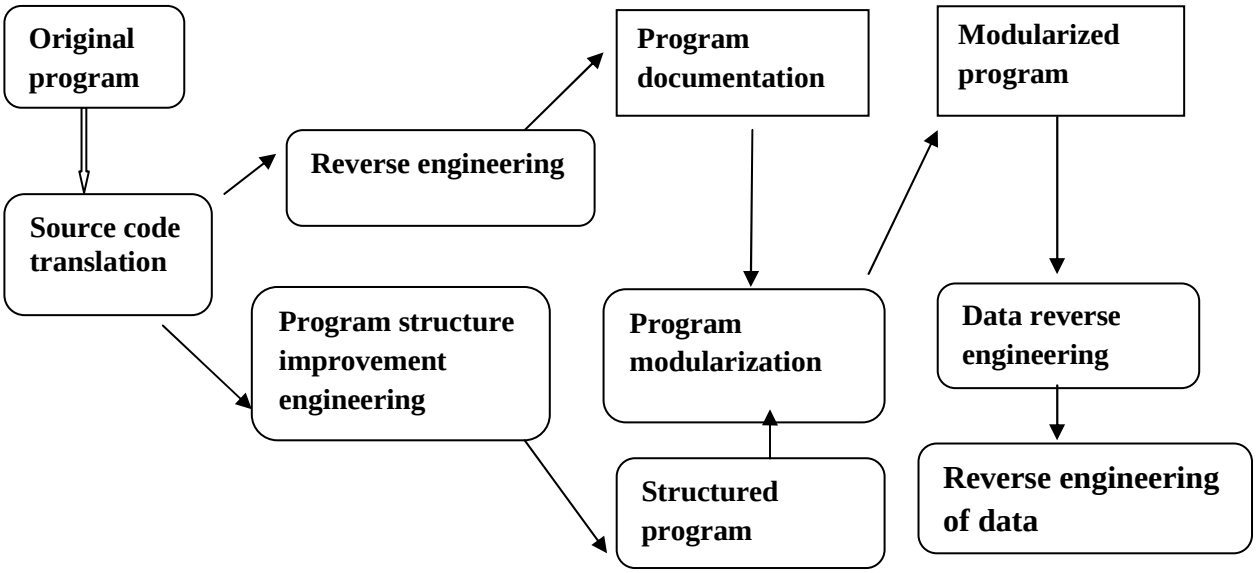

Fig.3: Illustrates a possible reverse engineering process.

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. It isn't possible, for example, to convert a system written using a functional approach to an object-oriented system. Major architectural changes or radical re-organizing of the system data management cannot be carried out automatically so involve high additional costs. Although reverse engineering can improve maintainability, the reverse engineered system will probably not be as maintainable as a new system developed using modern software reserve engineering methods. We are work on the reverse engineering of dialysis software.

VARIOUS TYPES OF HAZARDS OF SOFTWARE

The Software Hazard Analysis provides a detailed hazard evaluation of the system's software. Firmware to identify, determine software contributions to system safety analysis tools tend to be divided into deductive methods and inductive methods. Hazard analyses are performed to identify hazards, effects hazard and causal factors hazards. Hazard analyses are used to determine system risk and thereby ascertain the significance of hazards so that safety design measures can be established to eliminate or mitigate the hazard.

Analyses will be performed to systematically examination of the system, subsystem, facility, components and their interrelationships. There will be two categories of hazard analyses of *types* and *techniques*. Hazard analysis type defines an analysis category means design and technique defines a unique analysis methodology called fault tree analysis. The type establishes analysis timing, depth of detail, and system coverage. In general, there are several different techniques available for achieving each of the various types. a) *Conceptual design hazard analysis type (CDHAT)*; b) *Preliminary design hazard analysis type (PDHAT)*; c) *Detailed design hazard analysis type (DD-HAT)*; d) *System design hazard analysis type (SD-HAT)*; e) *Operations design hazard analysis type (ODHAT)*; f) *Health design hazard analysis type (HD-HAT)* and g) *Requirements design hazard analysis type.*

Hazard analysis type describes the scope, coverage intended to provide a time- or phase-dependent analysis that readily identifies hazards for a particular design phase in the system development life cycle. Since design that is more detailed and operation information is available as the development program progresses, so in turn more detailed information is available for a particular type of hazard analysis. The depth of detail for the analysis type increases as the level of design detail progresses. Each of these analysis types defines a point in time when the analysis should begin, the level of detail of the analysis, the type of information available, and the analysis output. The goals of each analysis type can be achieved by various analysis techniques.

The analyst needs to carefully select the appropriate techniques to achieve the goals of each of the analysis types. An important principle about hazard analysis is that one particular hazard analysis type does not necessarily identify all the hazards within a system; identification of hazards may take more than one analysis type (hence the seven types). A corollary to this principle is that one particular hazard analysis type does not necessarily identify all of the hazard causal factors; more than one analysis type may be required. After performing all seven of the hazard analysis types, all hazards and causal factors should have been identified; however, additional hazards may be discovered during the test program.

CONCLUSION

Reverse engineering improves the system structure, creates new system documentation and makes it easier to understand. Reverse engineering a software system has advantages over more radical approaches to system 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. A hazard analysis type defines the analysis purpose, timing, scope, level of detail and system coverage; it does not specify how to perform the analysis.

A hazard analysis technique defines a specific and unique analysis methodology that provides a specific methodology and results. There are seven hazard analysis types in the system safety discipline that, together help ensure identification and resolution of system hazards. There are over 100 different analysis techniques that can be used to satisfy the analysis type requirements and one particular hazard analysis type does not necessarily identify all the hazards within a system; it may take more than one type and usually all seven types.

ACKNOWLEDGMENT

Authors thank to UGC- Rajiv Gandhi National Fellowship for providing funds, e-consultancy of software's, and Gargik technologies, Mumbai for providing necessary facilities for research work.

REFERENCES

1. R. A. Converse and M. J. Bassman, Avionics Panel Symposium: Software Engineering and Its Application to Avionics, AGARD-NATO, Cesme Turkey, 1988, paper 8.
2. P.LClemens and W.T.Warner, A Perspective on System Safety Hazard Prevention, 1st Edn.1995.
3. C.A.Ericson, Hazard Analysis Techniques for System Safety, John Wiley & Sons, 2005.
4. B.E.Goldberg, System Engineering Toolbox for Design-Oriented Engineers," NASA Reference Publication 1358, December 1994.
5. V. Popovic,, B. Vasic and D.Curovic, Failure modes, effects and risks analysis – FMERA, Journal of Institute for Research and Design in Commerce & Industry,2008, 6(20) ,33-42.
6. J. Todorovic,: Maintenance Engineering of Technical Systems, Institute for Research and Design in Commerce & Industry and Faculty of Mechanical Engineering, Belgrade, 2006..
7. M.Rausand and A.Høyland, System Reliability Theory – Models, Statistical Methods and
a. Applications, John Wiley & Sons, New Jersey, 2004.
8. V. Narayan, Effective Maintenance Management: Risk and Reliability Strategies for Optimizing Performance, Industrial Press, New York, 2004.
9. V. Popovic, B. Vasic and N.Stanojevic, Contribution to development of new failure analysis methods, in: Proceedings of the 3rd World Congress of Maintenance, 2006, pp. 155-160.
10. S. Freiberger*, M. Albrecht and J. Käufl, Reverse Engineering Technologies for Remanufacturing of Automotive Systems Communicating via CAN Bus, Journal of remanufacturing 2011, 1:6
11. M.-A. D. Storey, F. D. Fracchia, and H. A. Müller. Cognitive design elements to support the construction of a mental model during software exploration. J. Syst.Softw., 1999, 44(3):171–185.
12. Richard Wettel and Michele Lanza. Visualizing software systems as cities. Visualizing Software for Understanding and Analysis, International Workshop on, 2 007, 0:92–99,
13. R. Keller, R. Shauer, S. Robitaille, and P. Pag´e. Patternbased reverse-engineering of design components. In Proc. of the 21st International Conference on Software Engineering, IEEE Computer Society Press, May 1999 pages 226–235.
14. P.Mendes, W .Sha, Ye K. Artificial gene networks for objective comparison of analysis algorithm. Bioinformatics 2003; 2(19Suppl):II122.
15. Salgado H, et al. RegulonDB (version 4.0): transcriptional regulation, operand organization and growth conditions in Escherichia coli K-12. Nucl Acids Res 2004; 32:D303.
16. M. Aluru, J. Zola, D.Nettleton and S. Aluru, Reverse engineering and analysis of large genome-scale gene networks, Nucleic Acids Research Advance Access 2012, 1-13.
17. T. Schaffter, D.Marbach and D.Floreano, GeneNetWeaver: in silico benchmark generation and performance profiling of network inference methods. Bioinformatics, 2011, 27, 2263–2270.
18. R.Nayak, M.Kearns and R.Spielman, Coexpression network based on natural variation in human gene expression reveals gene interactions and functions. Genome Res., 2009, 19, 1953–1962.
19. L.Mao, J. van Hemert, S. Dash and J. Dickerson, Arabidopsis gene co-expression network and its functional modules. BMC Bioinformatics, 2009, 10, 346.

*Correspondence Author: **Lalita M .Lokhande;** Department of Computer Science, Yeshwant Mahavidyalaya, Nanded-431601(India) Email: lalita.mv@rediffmail.com