Understanding Software Systems Using Reverse Engineering Technology

Muthu Dayalan
Senior Software Developer,Chennai, India

Abstract – **In the world of technological advancements, software systems hold a central position in the performance of software. Software engineering has therefore, been more reliable in the construction of stable software systems. However, the maintenance and analysis of the software systems has been neglected for a long time. In recent research, it has been illustrated that software maintenance is necessitated to provide end users with a more reliable experience. To solve issues in the software systems defects, reverse engineering is used to correct changes and restore possible loss of information. In this paper, the concept of proper balancing in software engineering is discussed. The software programming concept is also discussed and how it relates to reverse engineering. The concept and approaches of reverse engineering are discussed to show how software systems in need for proper maintenance. Six objectives of reverse engineering discussed in the paper reflects on why reverse engineering is more effective and affordable compared to making of new software systems.**

Keywords— **software engineering, reverse engineering, source code, software construction, software articraft.**

I. INTRODUCTION

Software engineering has been in existence since the 1968 and has become so siginificant in organizational development. This is evident due to the ever first software engineering conference held in Garnisch, Germany. In this case, software engineering refers to the application of enginering to develop software in a systematic way [1]. It is through software engineering that user needs are analyzed and designed through consturction and testing of the end user applications. Unlike simple programming, sogtware engineering has been used for larger and more complex software system making software systems critical for businesses and organizations. In the process of software development, software engineering tends to engelct aspecys such as maintenance and evolution leading to a software becoming less effective than its intended purpose. Reverse engineering is therefore introduced as a possible solution to program understanding and software analysis [2]. It has become an essential [art of development since software pioneers did not anticipatethat their software that they constructed in the 1960's and early 1970's would become modified years later. With cush software crisis existing, software systems crises have been minimized through reverse engineering. Some of the software that have been transformed include the telephone switching systems, banking systems, health systems, and pervasive computer vendor products. Health information, for instance, require to rapidly adopt to the new changes thus practitioners will be willing to support their software re-engineering process. In the process of re-engineering, it is likely that the historical concepts will be wiped off and replaced with more critical components of software systems. It is therefore important, for software developers to first learn and understand how software systems are impacted by the software reverse engineering technological processes [3]. In this research paper, the knowledge on reverse engineering, balancing act in the reverse engineering, program understanding via reverse engineering, approaches to reverse engineering, and an example of the Rigi Project.

II. NEED FOR BALANCING SOFTWARE ANALYSIS AND SOFTWARE CONSTRUCTION

The software developers are in a crisis for only concentrating more on software construction and not balancing between software construction and software analysis. In the late twentieth century, the software engineers have been concentrating on the software construction systems and negelcted software development and evolution [4]. Under software construction, some of the tools and methodologies that have been used include the programming languages, programming environments, and development of early phases in the software life cycle. As a result, there have been imbalances in software engineering education in both the industry and academia. As a result, there have been pressure to software engineers to be encouraged to have fresh creation and synthesis of concepts such as architecture, consistency, and completeness. There will always be new and software thus it will remain economical to maintain and ajust the existing one to meet the changes in its applications [5].

For a proper balance between software maintenance and construction processes, it is essential for users and developers to have knowledge of architectural concepts in larg software systems. Reverse engineering architectur include parts such as subsystem structures, layered structures, aggregation, specialization, and inheritance hierachies. To have software forward and reverse engineering, there are three dependent factors considered: existence of a life-cycle model, the presence of a subject system, and the identification of abstraction levels. It is worth noting that orderly life-cycle meodel exists for the software development process thus software models can be iterated with stages. Reverse engineering is, therefore, not focused on changing the sunbject system but rather to examining without changes or replication [6].

III. SOFTWARE PROGRAMMING THROUGH REVERSE ENGINEERING

There are several definitions offered on what reverse enginerring refers to. The basic concept is that reverse engineering is the process of analyzing a subject system to identify the system's components and their interelationships through creation of representation in the software systems to transofrm it to a higher a level of abstraction. The basoc step to reverse engineering is program understanding how reverse engineering works. Programmers use the programming knowledge, domain knowledge, and comprehension strategies to understand how a software system operates. In the worls of Brooks, the theory of domain bridging has descibed how programming process are constructed through mappings from the problem domain to the implementation domain [6]. Program understanding thus involves reconstruction of part or all the identified mappings. Morever, the programming process is cognitive and involves assembling of the programming plans. Resultantly, program understanding thus attempts to explain the patterns that match between a set of known plans and the source code of the subject software [7: 8]. The following figure illustrates on how reverse engineering looks like and how the related processes are transformational between or within abstraction levels.

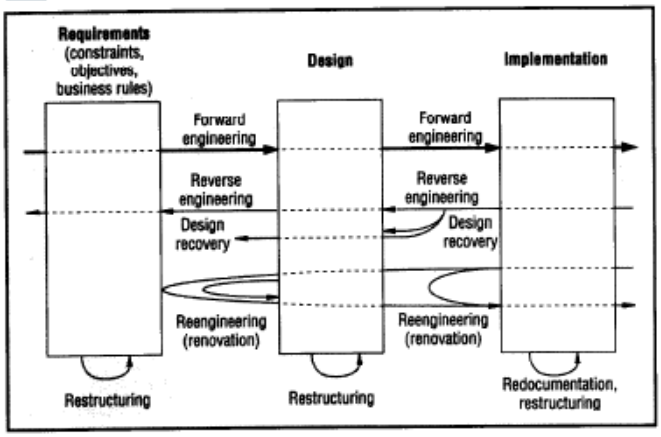

Figure 1. Relationship between terms. Reverse engineering and related processes are transformations between or within abstraction levels, represented here in terms of life-cycle phases.

Source: Chikofsky & Cross

IV. CONCEPTS OF REVERESE ENGINEERING

The common goal of reverese engineering is to extract information from exisiting software systems to better understand them. Reengineering consists of three stages, which include, reverse reengineering, restructuring, and forward engineering [3]. Reverse engineering involves two distinct phases. First step involves identification of the system's current components and captures their dependencies. Second phase discovers design information and generates system abstractions. The second phase is more interactiveand have more cognitive activities. During the reverse engineering process, it is critical that the source code is not altered despite additional information being generated. When making comparisosns, the business process re-engineering re-examines and streamlines how businesses work. Similarly, infromation technology is the key driver to reverse engineering and is therefore, often introduced into the workplace to simply automate old ways of doing business. Software-reengineering onvolves many risks. Before embarking on a significant re-engineering project, the goals must be very clear.

V. APPROACHES TO REVERSE ENGINEERING

Reverse engineering has many supporting aspects. In some instances, it may focus on features such as control flows, global variables, data structures, and resource exhanges. At a higher level, it tends to focus on features such as memory usage, unutilized variables, value ranges, and algorithimic plans. With the many revere and re-engineering tools available, the research on reverse engineering consists of many diverse approaches [8]. The three common approaches used are the formal transformations, pattern recognition, and the reuse-orinetned approaches. The pattern recognition approach seeks to find matching patterns and further involves defect filtering, syntatic cliches, user interface analysis, characterizing design decisions, function abstraction, information abstraction, and the graph parsing [9]. Under formal transformation approaches some of the main activities include: concept recognition amd transformation, and the least common abstractions. Finally, under the re-use approach, some of the re-use concepts used include: reuse-oriented software development, design recovery, and teleological maintenance. An example that illustrates on how software reverse engineering is being used is the Rigiproject, initated at the University of Victoria. Rigi project promotes reconstruction of the design of existing software is seen to be complex [8]. Rigi's framework primarily consists of a parsing subsystems thus have a structured representation of the desired software system [8].

VI. PURPOSE OF REVERSE ENGINEERING

The accomplishments intended form reverse engineering in a software system is primarily to increase the overall comprehensibility of the system for both maintenance and new development. There are six main objecives discussed and associated with the purposes of reverse engineering. Firstly, reverese engineering helps to cope with complexity. Software system developers must develop methods to efficiently deal with the shear volume and complexity systems. The channel and direction to control these attributes is having automated support. Reverse-engineering methods are combined with other tools to better deal with the small volume and complexity of systems. The second objective involves generation of alternate views with use of graphical representations that have long been accepted as comprehension aids [10]. However, creating and maintaining them continues to be a bottleneck in the process. The reverese engineering tools facilitate graphical representations from other forms to aid review and verification. The third objective is retrieval of lost information in software systems. Woth the continous evolution of large, long-lived systems, there is a

tendency to lose information about the system design. Modifications are not always reflected on the documentation of a software system, especially in the coding. Design recovery is introduced in reverse engineering to salvage whatever can be spared from the existing system [11]. The other objective of reverse engineering is to detection of side effects of a software system. Both risky and successive modifications are necessary and in some cases may lead to indeliberate ramificationsand side effects that impede a system's performance. Reverse engineering thus provide observational analysis leading to better evaluation in forward-engineering. Anaomalities can therefore be easily detected before software users identify them as bugs [14]. The fifth objective of reverse engineering is that reverse engineering synthesize higher abstractions. Methods applied to create alternative views transcends to higher abstraction levels. Clearly, the expert system technology will play a major role in achieving the full potential of generating high-level abstraction. The sixth major objective is that reverse engineering allows re-use which aids in the shift towardssoftware reusability even in a pool of large existing software components.

VII. SUMMARY

Software systems are vulnerable to being old after a proper construction process. It is critical for software developers in the softwware industry to deal effectively with issues of evolution and comprehensive software systems legacy. With the changing trends of construction of new software, reverse engineering has become useful in explaining how software systems operate in the day-to-day basis. Software engineering, reseath and construction must have major adjustments to effectively meet the needs of end users. In particular, more resources are required to be channeled to reverse engineering to have effective software systems.effective reverse engineering technologies is considered to have a significant impact on the maintenance and evolution of these systems [15]. An example of the Rigi project, shows how the approach to reverse engineering focus on critical software analysis. Software architecture is used to analyze the different operating environments. The software system structures are analyzed through identification, exploration, summarization, and evaluation of the environment in which a software structure lies within. Reverse engineering approaches used in software structures differ based on the exact need required to protect organizational development [10, 12, 14]. The role of the reverse engineering has been identified to include at least six objectives that promote effectiveness in the long-run. Abstractions in minds of software engineers require further implementation of software reverse engineering require to be strategically changes to ensure reliability and promotion.

REFERENCES

[1] Anonymous "Press Release: Perceptron Launches Smart3D(TM) Laser Scanning System Revolution in Automatic Reverse Engineering," *Dow Jones Institutional News*, 2015. Available: https://search.proquest.com/docview/2069556696?accountid=45049.

[2] R. Brooks. Towards a theory of the comprehension of computer programs. International Journal of Man-Machine Studies, 18:543- 554, 1983.

[3] S. Freiberger, M. Albrecht and J. Käufl, "Reverse Engineering Technologies for Remanufacturing of Automotive Systems Communicating via CAN Bus," *Journal of Remanufacturing*, vol. 1, *(1)*, pp. 1-14, 2011. Available:https://search.proquest.com/docview/1652978886?accountid=45049. DOI: http://dx.doi.org/10.1186/2210-4690-1-6.

[4] R. Perez-Castillo *et al*, "Reengineering Technologies," *IEEE Software*, vol. 28, *(6)*, pp. 13-17, 2011. Available: https://search.proquest.com/docview/898842162?accountid=45049.

[5] D. Seng, "REVERSE ENGINEERING THE NEW REVERSE ENGINEERING PROVISIONS IN THE COPYRIGHT (AMENDMENT) ACT 2004," *Singapore Journal of Legal Studies*, pp. 234-245, 2005. Available: https://search.proquest.com/docview/222725442?accountid=45049.

[6] S. Rugaber and K. Stirewalt, "Model-driven reverse engineering," *IEEE Software*, vol. 21, *(4)*, pp. 45-53, 2004. Available: https://search.proquest.com/docview/215843843?accountid=45049. DOI: http://dx.doi.org/10.1109/MS.2004.23.

[7] R. Behling, C. Behling and K. Sousa, "Software re-engineering: concepts and methodology," *Industrial Manag, ement & Data Systems*, vol. 96, *(6)*, pp. 3-10, 1996. Available: https://search.proquest.com/docview/234917397?accountid=45049.

[8] H. A. Miller. Rigi -A Model for Software System Construction, Integration, and Evolution based on Module Interface Specifications. PhD thesis, Rice University, August 1986.

[9] E. J. Chikofsky and J. Cross II H., "Reverse Engineering and Design Recovery: A Taxonomy," *IEEE Software*, vol. 7, *(1)*, pp. 13-17, 1990. Available:https://search.proquest.com/docview/215834539?accountid=45049. DOI: http://dx.doi.org/10.1109/52.43044.

[10] G. Canfora, M.D. Penta, and L. Cerulo, “Achievements and Challenges in Software Reverse Engineering,” Comm. ACM, vol. 54, no. 4, 2011, pp. 142–151.

[11] H.M. Sneed, “Estimating the Costs of a Reengineering Project,” Proc. 12th Working Conf. Reverse Eng., IEEE CS Press, 2005, pp.111–119.

[12] H. M • uller and J. Uhl. Composing subsystem structures using (k; 2)-partite graphs. In Proceedings of the 1990 Conference on Software Maintenance (CSM '90), (San Diego, California; November 26-29, 1990), pages 12{19, November 1990. IEEE Computer Society Press (Order Number 2091).

[13] R. Mall, Fundamentals of software engineering. PHI Learning Pvt. Ltd, 2018.

[14] S. Senthivel, S. Dhungana, H. Yoo, I.Ahmed, and V. Roussev. "Denial of Engineering Operations Attacks in Industrial Control Systems." In *Proceedings of the Eighth ACM Conference on Data and Application Security and Privacy*, pp. 319-329. ACM, 2018.

[15] J. Duchêne, C. Le Guernic, E. Alata, V. Nicomette, and M. Kaâniche. "State of the art of network protocol reverse engineering tools." *Journal of Computer Virology and Hacking Techniques* 14, no. 1 (2018): pp.53-68.