**Journal**

Justin Starr

Department of STEM

CS 410 – Software Reverse Engineering

Dr. Philomena Ogoh

September 22, 2024

**Journal**

The following is a short analysis of what software requirements engineering is, why it’s an important part of the software development lifecycle, how the approach of software reverse engineering differs from the approach of software requirements engineering, and what my thoughts are on the proposed new integrated approach of round-trip engineering and its impact on the computer science field.

Software requirements engineering focuses on taking software requirements and turning them into a description of what the required software is, what its performance parameters are, and its configuration through an iterative process definition, tradeoff studies, conducting analysis, and prototyping (Fahmi & Choi, 2007). It includes defining and analyzing requirements including functional and non-functional specifications from common sources, such as customers, buyers, users, operators, domain experts, stakeholders, etc. (Fahmi & Choi, 2007). It is essentially a strict and systematic approach to defining, creating, and verifying the requirements for a software system while managing and ensuring the needs and expectations of stakeholders for a software system are met.

The reason why software requirements engineering is an important part of the software development lifecycle (SDLC) is because oftentimes, software reverse engineering and requirements engineering deal with improving legacy software/code. This is considered to be a part of the maintenance phase of the SDLC and is a critical aspect of ensuring systems stay up-to-date, especially as the needs of businesses change. More often than not, when dealing with legacy software, there is a lack of documentation, which then requires software/systems to be reverse-engineered to determine what the original needs were that the software or system was meant to serve. By recovering requirements from reverse engineering and incorporating them into the requirements phase of the SDLC, it is possible to have better requirements elicitation and a better understanding of what portions of software or systems may be redundant or possibly no longer needed, and what portions can be retained and re-used (Fahmi & Choi, 2007).

Software reverse engineering is essentially the process of analyzing a specific system and identifying its components to determine what their inter-relationships are. This enables engineers to have representations of the system in various forms or at a higher level of abstraction and is a way to get back missing requirements or design specifications of a system (Fahmi & Choi, 2007). The reverse engineering process is divided into four main parts, context parsing, component analysis, design recovery, and design reconstruction, essentially having a backward approach to creating the necessary documentation for a system and its components (Fahmi & Choi, 2007). Software requirements engineering, on the other hand, has a more forward approach to documenting a system and its components. The process involves creating documentation during the design and development phases of the SDLC by conducting requirements elicitation, analysis, specification, and validation (Fahmi & Choi, 2007). This helps ensure that all problems are realized concerning requirements, which facilitates the development of a better product, and helps ensure the maintainability of systems.

My thoughts on the proposed new integrated approach of round trip engineering are that it is a necessary approach, meaning that it is essential to combine requirements engineering with reverse engineering before the architecture and design phase of an operational system, particularly in the case of software reverse engineering where, in the past, the primary focus was on design recovery where the information was used primarily for architecture and design. The adoption of the new integrated approach of round-tripping can contribute by incorporating a significant amount of requirements from a legacy system to a new system, it becomes possible to determine whether any requirements are missing or not, or if they are lacking in any way, it also becomes easier to identify whether requirements need changing, and previous changes to requirements can be more easily identified (Fahmi & Choi, 2007).

**References**

Fahmi, S. A., & Choi, H. J. (2007). Software Reverse Engineering to Requirements. *2007*

*International Conference on Convergence Information Technology (ICCIT 2007)*, 2199-2204. doi: 10.1109/ICCIT.2007.228.

GeeksforGeeks. (2024, July 24). *Requirements Engineering Process in Software Engineering*.

<https://www.geeksforgeeks.org/software-engineering-requirements-engineering-process/>