**3-1 Journal**

Mohamed Saleh

Southern New Hampshire University

CS 410: Software Reverse Engineering

Prof. Mark Babock

January 25,2025

Software requirements engineering is a systematic and strict approach to definition, creation, and verification of requirements for a software system is known as requirements engineering. To guarantee the effective creation of a software product, the requirements engineering process entail several tasks that help in understanding, recording, and managing the demands of stakeholders. Software requirements are mainly divided into functional and non-functional; functional requirements define what actions or functionality the system should provide or the user able to do. During the functional gathering process system analysts conduct personal interviews with potential users, verifying and validating the requirements. Prototyping and surveys also contribute to clarifying the quality of the requirements. In the other hand non-functional requirements are mostly related eco-system of the system like scalability, security, disaster recovery, backup planning.

Software requirements are the backbone of building software and the main driver for all the needed efforts and technological choices, building a software for 100K users is away different than building a software for 100M users, the considerations and the cost for building the software for 100M users is extremely high and involves multiple teams to handle different aspects of the system from planning, development, testing, deployment, documenting, deployment and infrastructure requirements. Inherently software requirements are a major factor in software development.

Software reverse engineering is the process of analyzing a subject system to identify the system’s components and their interrelationships and to create representations of the system in another form or at a higher level of abstraction, it’s the way to get back the requirements from a system. The purpose of software reverse engineering is to understand how systems work at a deeper level and aid in maintenance of legacy systems. The process of software reverse engineering involves multiple activities to inspect the system and break it down into pieces and understand the main functions, behaviors, rationale and the structure. Interestingly, we currently have many tools and frameworks to inspect different systems from network to low levels. Reverse engineering process is mainly divided into four steps Context Parsing, Component Analysis, Design Recovery and Design reconstructing. The outcome of each process cumulatively can draw out the design model of the system of study.

The key characteristic of round-trip engineering is the ability to synchronize existing artifacts that evolved concurrently by incrementally updating each artifact to reflect changes made to the other artifacts. I think the impact of RTE on computer science is significant through applying full or partial directional RTE and iterative development. However, there should be standardization for the process due to the vast number of compilers, linkers and programming language versions for example there are multiple flavors for the same technology.

**References**

*GeeksforGeeks. (2024, July 24). Requirements Engineering Process in Software Engineering.* [*https://www.geeksforgeeks.org/software-engineering-requirements-engineering-process/*](https://www.geeksforgeeks.org/software-engineering-requirements-engineering-process/)

*Choi, H.-J., & Fahmi, S. A. (2015). Software reverse engineering to requirements | IEEE conference publication | IEEE xplore. Software Reverse Engineering to Requirements.* [*https://ieeexplore.ieee.org/document/4420580/*](https://ieeexplore.ieee.org/document/4420580/)

*Wikimedia Foundation. (2024, July 10). Round-Trip Engineering. Wikipedia.* [*https://en.wikipedia.org/wiki/Round-trip\_engineering*](https://en.wikipedia.org/wiki/Round-trip_engineering%20)