**3-1 Journal**

Winnie Kwong

Southern New Hampshire University

CS-410 Software Reverse Engineering

Professor Priest

May 26th, 2024

**Defining Software Requirements Engineering (SRE)**

SRE helps understand software development to ensure expectations are met while correcting mistakes to prevent future costly slipups. According to Jędrzejko, the process of SRE defines, documents, and maintains requirements. (Jędrzejko, 2023, para 1).  Determining what the software needs to do and how it should function enables developers to build successful software and guarantee everyone is on the same page. SRE involves crucial steps such as elicitation, analysis, specification, and verification to make the software as accurate and sufficient as possible.

**Purpose of SRE in Software Development Life Cycles (SDLC)**

SRE has many benefits within SDLC, such as project management, coding, and testing. During the design life cycle, SRE helps build suitable projects for developers. Clear documentation establishes the team's goals and scope while avoiding miscommunication to align with the stakeholders' expectations. SRE can also benefit when coding by detecting contradictions in the solution. Developers developing the correct system architecture can mitigate problems that may occur in the future. Khan states that during the testing phase, software requirements engineering is used to identify and correct defects and to provide confidence that the product meets those requirements. (Khan, 2019, para 15). With requirements, testing throughout the life cycle can ensure all of the software's functionality works in the test cases.

**Comparison of SRE and Reverse Engineering**

Although SRE and reverse engineering have similarities in maintenance, there are differences, such as their approaches and goals. Both SRE and reverse engineering contribute to the overall maintainability of the software. While SRE provides clear instructions for understanding the program, reverse engineering can provide similar requirements for understanding the structure and design of the program. On the other hand, SRE must be accomplished at the beginning of the SDLC. In contrast, reverse engineering can only start their approach once the software exists. SRE can only provide instructions on hold to build a product that meets stakeholders' expectations. At the same time, reverse engineering requires an existing system to break down to fully understand the system's design, structure, and functionality. Furthermore, SRE aims to determine what and how the product will function, while reverse engineering seeks to find ways to create or improve a similar product.

**Impact of Round-Trip Engineering**

Round-trip engineering helps maintain software components to eliminate inconsistencies when information duplicates throughout various sources. It is a benefit for computer science fields because it provides better efficiency and communication while reducing errors that can lead to bugs and defects in the software. According to Fahmi et Choi., other ways round-trip engineering can impact computer science can be helpful because of its contribution to many legacy system requirements for a new system. (Fahmi et Choi, n.d., p. 2203). By streamlining requirements analysis, round-trip engineering can improve accuracy and traceability, making it easier to understand the new system.

**References**

Fahmi, S. et Choi, H*. Software reverse engineering to requirements*. (n.d.). IEEE Conference Publication | IEEE Xplore. <https://ieeexplore-ieee-org.ezproxy.snhu.edu/document/4420580>

Jędrzejko, K. (2023, July 6). The importance of software requirements engineering in IT projects. *Software Mind*. <https://softwaremind.com/blog/software-requirements-engineering-the-driving-force-behind-successful-and-efficient-it-projects/>

Khan, R. (2019, December 6). Software Requirements Engineering: What, Why, Who, When, and How. *Medium*. <https://medium.com/@mrummanhasan/software-requirements-engineering-what-why-who-when-and-how-bcf5fa729e3b>