**3-1 Activity: Journal**

Dylan Jeffery

SNHU

CS-410

Brian West

1/22/2022

This week we were asked to reflect on a few things, primarily aspects of requirements engineering. To begin we must first define what requirements engineering means. Requirements engineering often “deals with transforming software requirements into a description of required software, performance parameters, and configuration through iterative process definition, tradeoff studies, analysis, and prototyping,” (Fahmi, Choi). Through this process we home to recover requirements from a reverse engineered legacy system. In integrating this outcome in the requirements phase of a software’s life cycle, we hope to achieve better requirements elicitation, and a better understanding of what is usable/beneficial, and what is redundant. The common sources of software requirements are customers, buyers, acquisition agencies, users, operators, domain experts, system analysts, system specifications etc.

Although reverse engineering and requirements engineering are separate processes, combining them can provide a lot of value. The main purpose is that when round-trip engineering a legacy system integrating requirements and reverse engineering can give us a better understanding of the requirements for a new system. Integrating reverse engineering earlier in the life cycle phase of development, we can sometimes gain a better understanding of the requirements and in turn a better outcome.

The four activities associated with requirements engineering are elicitation, analysis, specification, and validation. Reverse software engineering usually consists of context parsing, component analyzing, design recovery phase and the design reconstruction phase. Reverse engineering typically examines and analyzes source code, whereas requirement engineering often works off information gained by the reverse engineering process.

The proposed approach of round-trip engineering has many expected advantages. It allows us to pull many requirements of the legacy system, while also acting as a supplement in the requirements stage of the roundtrip engineering. Our proposed integration allows us to easily check if any previous requirements are missing or not as well as which requirements may be lacking. Through this model we can trace if requirements have been changed or are changing, as well as tracing out useless requirements. The benefits of an approach like this are important to consider and moving forward, hopefully the future brings more research into this topic, and further refinements to the methodologies proposed here.

**Sources:**

S. A. Fahmi and H. Choi, "Software Reverse Engineering to Requirements," 2007 International Conference on Convergence Information Technology (ICCIT 2007), 2007, pp. 2199-2204, doi: 10.1109/ICCIT.2007.228.