Generally, when we talk about software requirements engineering. We refer to a multi-step process that includes but may not be limited to identifying, documenting, analyzing and managing the needs and constraints of software. Usually, the process involves interacting with

customers or stakeholders which help to outline the functional and non/functional characteristics of the software. The main goal of software requirements engineering is to ensure that the software meets the requirements the customer/stakeholders requested.

The importance of software requirements engineering on the software development life cycle is clear. In order to meet the stakeholders needs and requirements, this is a step that must be taken. This step helps the software development team understand what problem the software is trying to solve. Communication with the stakeholders is constant, thus, expectations are at a glance, since there is always a communication channel open with the stakeholders. Additionally,

risks can be noticed earlier which in turn increases the efficiency of software development.

So, how do the software requirements engineering, and the software reverse engineering defer? Well, the main difference is that while reverse engineering focuses on analyzing and understanding previously created software, requirements engineering focuses on similar aspects on the opposite end of the process, the beginning. Software engineering is majorly focused on defining what a system can do per the requirements. Reverse engineering focuses on documenting how a software system was created, while the counterpart focuses on created the guidelines on which o software is built upon.

Enters integrated approach of round-trip engineering. This concept can be very beneficial if done correctly. There could be a potential increase in efficiency if somehow the requirements and the design of the system are in sync based on automated propagation. If the requirements and the design of the system are in sync, traceability would be easier. Although all of this sounds fantastic in paper, round-trip engineering is quite complex, and the adoption is not as significant as the other options.

Stefan JetchickCurrently exploring ideas for a General-Purpose Systems Programming Language with the underlying Object Model of C++. (2016, August 25). *Round Trip Engineering for dummies: A proposal*. Modeling Languages. Retrieved January 29, 2023, from https://modeling-languages.com/round-trip-engineering-uml/