Chris Gollnick

CS410-T3717

Journal Software Requirements Engineering

21 January 2022

Define:

Software requirements engineering is a procedural process of analyzing the requirements of the software generally derived from the client and transforming them into a tangible listing of utilizable metrics. Such metrics may include require software, parameters for the performance of it, and configurations that will extend through the iterative process.

Purpose:

Software requirements engineering becomes an important part of the software development life cycle as the process of it and the outcome it produces helps identify the needs of the product, the changes that need to be tracked, and adds to if not creates the process for monitoring the progress and organizability during the life cycle. Tasks such as Software Requirements Elicitation help identify the actual needs of the product extrapolated from what is said by the client. This ensures the requirements that are truly needed are identified and further ensures that those requirements are met. The requirements analysis helps with the tracking process by creating a method of checking the actual requirements of the software against external factors to determine issues that exist and may arise as well as to understand why they do. In Requirements Specifications the identified requirements are then formalized so the client can then agree on what is being created and what it will and will not do. This ensure the design and development phases of the life cycle are then focused on the agreed needs and not unnecessary or unwanted features that may take away time and resources from more important ones.

Comparison:

While both Software Reverse Engineering and Software Requirements Engineering both may address the requirements of a software product, they attack it at two different angles. Reverse engineering will look at an existing system and attempt to work backwards to identify what the requirements were when the software was being built. In essence the process aids in helping to maintain the product through its life after launch by identifying what was originally required and how it behaves now and how to document and resolve it. While requirements engineering focuses on the other end from deriving what a client wants the end-product to do before development begins. It then aids and tracks the progress as it is created before launch.

Impact:

Integrating the process of reverse engineering into the earlier parts of the software development life cycle seems like a valid idea. While the requirements engineering pushes the product forward, reverse engineering used as a way to check what is done from the opposite angle can give better coverage. It may highlight issues with requirements further down the process that may occur that requirements engineering can not identify or would not see in the maintenance phases. The ability to have a way to see how adding functionality, usage, and integrations post launch can be critical to the acceptance and usage of the product once it’s in the end-users’ hands. If we fail to see a critical error with adding a potential feature after launch that causes the product to cease to work or result in heavy downtime, then the client will see the product as flawed and could potentially switch to a different software or development/engineering firm as a result of such.