Isabella Boone

Andrew Januszko

Dr. Wellington

SWE 415

13 February 2021

Reeves Rewritten

When comparing conventional engineering and software engineering, their differences are minimal. Both require rigorous planning, prototyping, and testing before their completion. Both must build and test models and revise as needed. In either case, before production, engineers must redesign and refactor their projects before finally committing to a build.

A build is when a project is provided to a third party to handle the construction. For software engineers, the cost of building a project is virtually free. Code can be compiled within seconds or hours using a compiler and a linker, allowing the software engineer to test their code, look for flaws, and tweak their design.

Unlike building software, projects in conventional engineering can take several years and cost millions of dollars, but there is more to it than cost. In some ways, these build times do exist in the form of software like Solidworks and Revit. These offer relatively fast 'build' times, allowing engineers to make virtual copies of their projects and run structural analysis simulations on them. When comparing the two concentrations, they have the same requirements, but the weight is not the same for each.

If both concentrations are relatively similar, then why does software engineering draw criticism? For conventional engineering, anyone can design a bridge, but making sure that the bridge is functional, well tested, and safe is the hard part. Compared to software engineering, it can be hard to design code depending on the language used and the end goal, but building the code is easy because the software does it for you. In the end, the cost of building a project depends on the toolset that either concentration decides to use.

Jack Reeves faced a lot of criticism for a paper on a similar topic, mainly for his general use of the word "design." He states, "software design is relatively easy to create," but later on states, "software design is a difficult and error-prone process." While he uses the same word, he covers three completely different aspects of software design. Those three intentions being the physical process of writing code, testing and debugging, and building a program. A secondary topic covered by Reeves includes the effect of object-oriented programming on the discussion.

Recently, there has been an increase in object-oriented languages available to the general public. These languages make it easier to design software since data and code reside in an "object," providing more developers the ability to express high-level information. The availability of object-oriented languages, plus added features like strong type checking and better error detection, allows for a better engineered design.

Ultimately, software engineering should be treated similarly to conventional engineering. While both have different approaches to what a build is, they have the same end goal in mind; releasing a product, be it programs or homes. The relative ease of building in software engineering makes it easy to dismiss, but tools exist in the modern world that removes many of the pains related to conventional engineering. While Jack Reeves agreed that software engineering should be treated similarly to other concentrations, his mention of object-oriented languages offers an optimistic future to the development processes for both concentrations.