Software Engineering Homework 2

Jude Jang

In “No Silver Bullet,” Frederick Brooks likens the rapid advances made in hardware to a silver bullet and states that fundamentally a silver bullet for software is unlikely. However, there are certain significant advances in software and technology that attempt to address certain problems.

One advancement in software engineering that could be considered a “silver bullet” are the invention of IDEs. Integrated development environments were designed to increase productivity by providing many development tools and features in one environment and reducing setup time. Although Frederick Brooks states that artificial intelligence is not a “silver bullet” in his paper, at the very least it is hard to disagree that artificial intelligence is a powerful technology. Artificial intelligence is used for a variety of things, but overall, I would say that it is used to help people in some form or fashion. An upcoming, but controversial “silver bullet” are autonomous cars. Autonomous cars aim to reduce overall traffic, reduce traffic collisions, and reduce energy usage. However, it will still be a while before autonomous cars will be commercially available to the public.

Frederick Brooks believes that the reason why there has not been a significant advancement in software engineering compared to advancements made in hardware are because we have been mostly addressing accidental complexity, but not essential complexity. In a very simplified manner, accidental complexity is unnecessary complexity caused by human error, where essential complexity is an inherent property of a problem.

In the technologies I listed above, they all respectively attempt to reduce accidental complexity. Having your development tools spread out instead of having them in one environment creates unnecessary time the programmer has to spend swapping between each development tool when given a task. One example of artificial intelligence showcasing how people create accidental complexities is AlphaGo, a machine learning computer program that beat one of the world’s best Go player back in March 2016. One could say that the reason why AlphaGo won was because it was optimized to make less mistakes/moves or have fewer accidental complexities. Autonomous cars aim to reduce accidental complexities by reducing the number of traffic collisions due to human error.

With all technologies however, essential complexity still remains. Although IDEs reduce accidental complexities, we still rely on development tools to develop programs. Artificial intelligence is used aid us in solving problems, but difficult problems are difficult because we are limited in the way we approach solving problems. One of the goals of autonomous cars are to reduce traffic, but who is to say that cars are the best form of transportation and that there is not a better way?

ULS systems are inherently complex due to the sheer scale of different variables within the system. Using transportation as an example, some essential complexities within this system are environmental impacts due to pollution, the loss of available public space due to cars needing parking space, and overall traffic congestion. As cars are our main form of transportation car emissions are unavoidable until there is a less harmful form of transportation that becomes commercially accessible. As more urban areas arise, so will the number of cars and the need of parking space. As population density increases in urban areas, so does traffic congestion as more vehicles become a part of traffic.

Potential accidental difficulties posed by ULS system are the increased frequency of the types of problems that smaller scale systems would already have and potentially new types of faults. Due to the scale of ULS systems, the number of failures per unit of time also increases and becomes a continuous problem.

These accidental difficulties can be mitigated by designing ULS systems to be able to contain the effects of accidents and if possible, set warnings in place before the accidents occur or become too big. ULS systems should also be designed to limit how much of the system can be affected by accidents and have higher fault tolerance as well as implementing new strategies for fault tolerance.

**References**

[1] F. Brooks Jr., “No Silver Bullet: Essence and Accidents of Software Engineering”, IEEE Computer, Vol. 20, Issue 4, April 1987, pp. 10-19

[2] “Ultra-Large Scale Systems”, The software Challenge of the Future, SEI, 2006