Andrew Cunningham

CSC505-Assignment 1

Critical Thinking Discussion:

1. Why does it take so long to get software finished?

The primary driver for the time commitment to developing software is the scale and complexity. A very simple script, like the Merge Sort that I submitted, took around an hour to design, develop, and test. Compare that to an industrial complex like Netflix, which is a software that will never be finished and has cost billions.

This makes scale the primary driver for the time commitment. The more complicated the product, the larger number of man hours required to produce it. Scale and complexity are related, at least for the purposes of impact to overall development time, as I’m considering man-hours.

In his article, “Why does Software Cost so Much”, Robert Stoddard uses this concept called “causal learning” to analyze why costs balloon so much in both time and money. He attempts to understand the true factors of cost, and not just make simple correlations, such as more expensive developers driving up costs. Stoddard ended up finding correlations between ballooning requirements, underestimation, and complexity as being several of the driving factors. (Stoddard, 2018).

Erik Bernhardsson studied statistical models related to the time required to write software. His research found that it’s very easy to underestimate time commitments, and most tasks take 65% longer than promised, with the upper 99 percentile often being a factor of 10 higher. (Berhardsson, 2020). My take away from this is that it’s normal for software to take longer than expected to complete, and we should be more generous with out promised deadlines.

1. Why are development costs so high?

The first thing that comes to mind for costs is the time commitment. If a project takes 65% longer it might seem logical to conclude that it’s also 65% more expensive, but this is far from the truth.

Hardware costs are also starting to make a major impact on the overall costs, especially in AI development. IBM estimates that the cost of compute is expected to rise by 89% between 2023 and 2025. (Brodsky, 2024). This is just one example of why software can cost so much to develop, we can also see the cost in the complexity and scale of major tech company data centers, which continue to get larger to support a larger role. The cost of compute has gotten to the point where the chips are more valuable than the engineers sometimes. There are a few articles around the internet that suggest AI company startups are struggling to acquire talent not because of salaries, but because the company cannot acquire enough chips to keep satisfy the developer’s thirst for compute. (msmash, 2023)

1. Why can't we find all errors before we give the software to our customers?

An article published in the computer journal deals with this discussion, where the authors cite previously discussed development costs and growing complexity as being main factors. They also stress the scale that comes out of complex interactions, testing limitations, and dynamic environments. (Zhang, 2016)

Debugging before launch is a cost-benefit analysis with logarithmic results. The only way to flesh out a bug is to perform the action that produces the bug, patch the bug, then retest. At first it’s easy to find and patch bugs, but as they get patched it takes longer and longer to find more bugs. This means that there is diminishing returns. Managers must make risk decisions regarding the acceptable amount of bugs to ship with a product, as achieving zero is an infinite task.

Additionally, a live software is also used by sometimes many orders of magnitude more people than the development and QA teams. This means that bugs won’t even be revealed until a user creates the exact environment to create the bug, which wasn’t examined during testing due to complexity and scale.

1. Why do we spend so much time and effort maintaining existing programs?

As a Submarine Officer in the US Navy, I have thought about this question a lot. Why are we still using systems from the 1970s to communicate and conduct strategic nuclear warfare? The simple answer is it works. We put the time and effort into developing these simple systems decades ago and they have stood the test of time. Why mess with something that isn’t broken? We now have many years of debugging time through trial and error, and replacing these systems would incur not only the previously discussed costs, but also incur that much more debugging time.

1. Why do we continue to have difficulty in measuring progress, as software is being developed and maintained?

Based on my research, I believe it’s mostly based on the human factor. (Bernhardsson, 2020). We, as people, want to stay optimistic and provide the client with an answer that sounds good and is more likely to earn us trust and business. We cannot foresee all of the complications, and if we promise a timeline that feels to long, then the customer might go with another bidder. There are unknown unknowns with every project that take time, and the development process is a constant struggle between perfection and outputs.

References:

Stoddard, R. (2018, April 16). *Why does software cost so much?*. SEI Blog. https://insights.sei.cmu.edu/blog/why-does-software-cost-so-much/

Bernhardsson, E. (2020, April 19). *Why software projects take longer than you think: A statistical model*. Erik Bernhardsson. https://erikbern.com/2019/04/15/why-software-projects-take-longer-than-you-think-a-statistical-model.html

msmash. (2023, April 7). *AI developers stymied by server shortage at AWS, Microsoft, Google*. Slashdot. https://tech.slashdot.org/story/23/04/07/1618235/ai-developers-stymied-by-server-shortage-at-aws-microsoft-google

Zhang, T. (2016, May 5). *https://ieeexplore.ieee.org/document/8213108*. Oxford Academic. https://academic.oup.com/comjnl/article-abstract/59/5/741/2568647?redirectedFrom=fulltext&login=false