Ryan Filgas
CS300 – Assignment 1
Chris Gilmore
10-13-2021

1. What is the most important difference between generic software product development and custom software development? What might this mean in practice for users of generic software products?

Software product development is for a broad set of users: anyone who wishes to buy them. Custom software development is made for a single company or individual to fit their needs. In practice this means that users of generic software have to learn how to adapt the software to their needs, and they may not have much pull when it comes to getting new features. Users of generic software may have to settle for available functionality where custom software is built specifically for the users purpose.

2. What are the four important attributes that all professional software should possess? Suggest four other attributes that may sometimes be significant.

Software must be maintainable, so easy to adjust and flex to the customer's needs. Software must be dependable (includes security), so it doesn't frequently break and isn't easy to compromise, or if it breaks does so quietly. Software must be efficient, not wasting system resources unnecessarily. Software must be acceptable to the user. Four other factors may include cost efficient, proven (used by other industry partners), ease of installation (it shouldn't slow down business for very long and ideally can be installed over night, or switched instantly), low maintenance (it might not be good software if it requires frequent maintenance due to changing needs, and too many needs weren't anticipated). All these additions might fall under "acceptable" as a catch all.

3. Explain why the fundamental software engineering principles of process, dependability, requirements management, and reuse are relevant to all types of software systems.

Process is relevant to all software systems because with process come things like dependability and performance. The risk of not working with a set thoughtful process is ending up with rework that could eclipse the original cost of the original software development process. Process models include waterfall, incremental development, and re-use oriented development.

Dependability is relevant to all software systems because for obvious reasons the software just has to work. If it frequently stops or breaks, it's causing problems instead of resolving them. This doesn't make for good software.

Requirements management is the initial building block on which the rest of a project leans. Without good requirements gathering and management a project has the potential to have much rework regardless of what that project is. Software needs to perform specific functions that are intentionally collected.

Software reuse is relevant to all software systems because it reduces overhead. In addition, it's the dominant approach for constructing web based systems. Work can be done quickly, and different components and be swapped out faster than redesigning software from scratch.

Ryan Filgas
CS300 – Assignment 1
Chris Gilmore
10-13-2021

4. Explain why change is inevitable in complex systems and give examples (apart from prototyping and incremental delivery) of software process activities that can help predict possible changes and make the software being developed more resilient to change.

Change is inevitable in complex software systems because customer needs can change over time, components may become outdated, and there's more chance of error the more complex software becomes. Software must be developed in a flexible way to minimize maintenance costs, and requirements gathering should not only figure out what the customer needs now, but leave room in the software for changes to to be easily made. Developing to allow for re-use is one way to ensure that many of the software components will carry through to later iterations of the software rather than being scrapped for a new build. The three components of the process improvement cycle to change, measure, and analyze are also activities to help predict possible changes. While future changes can't always be predicted, making flexible software can help manage potential issues.

5. Historically, the introduction of technology has caused profound changes in the labor market and, temporarily at least, displaced people from jobs. Discuss whether the introduction of extensive process automation is likely to have the same consequences for software engineers. If you don't think it will, explain why not. If you think that it will reduce job opportunities, is it ethical for the engineers affected to passively or actively resist the introduction of this technology?

Process automation is unlikely to cause much change in the field of software engineering for a few reasons. The main reason is that we haven't built a program that can read other programs and tell us what the program does. If a program can't know what another program does, the possibility that software development could be automated is unlikely. In addition software is constantly changing, and automation requires processes to be static and relatively predictable. This isn't to say that some peoples software jobs can be automated, of course they can. I think the more relevant danger is that we write enough useful software that there isn't a whole lot left to write such that if you can think of a useful software, it's already been done, and it requires little maintenance. This possibility though seems to be either in the very distant future, or unlikely to happen. There are also many types of software that wouldn't necessarily be automatable. Embedded systems for example are custom designed for every new device and invention. Are we going to stop inventing? I don't think so.

The question of whether to resist automation is easy to answer. If engineers don't work, they don't get paid. In that respect if they have conviction in this regard, then they must be willing to leave their jobs. Automation isn't inherently unethical, and I don't see an ethical reason to resist progress in this regard. What we do with the people that get displaced is more important. Eventually there will be less and less jobs for humanity to do. If we take care of each other and try to progress as a species rather than leaving those displaced to automation out to dry, then I believe most sensible people would agree that automation isn't only ethical, but preferred. Self driving cars for example are set to soon decimate the taxi and rideshare industries, and irrevocably change the way vehicles are bought and sold in addition to the concept of ownership over transportation. This will probably be one of the largest market disturbances humanity has ever seen, and it's going to be a moment where we as a people will have to decide how to take care of our own in the age of automation.