CS 362 Homework 1, Lyell Read

QPM.1: Briefly describe the fundamental differences between project-based and product-based software engineering.

At a basic level, project-based engineering is engineering that revolves around a customer-performer relationship. The customer gets to set the goals that they want for the software that they request, and the performer attempts to meet those goals. This process is typically cyclic, in that it involves a feedback system where the customer can understand completed progress and re-evaluate the requirements they set to the programmer. This model is typically implemented in scenarios where someone of a higher level of authority (customer) is resting software to be written by someone that likely works for them (programmer).

In contrast, project-based engineering takes place entirely at the developer level: software developers identify a market opportunity, and pursue the production of software to capitalize on that opportunity or need. In this process, the software engineers are responsible for all the steps, from identifying the opportunity, to developing the requirements / goals / features for the software, to implementing that software, and finally to checking that the implementation in fact satisfies the initial goal. I envision this as much more of a lean / start-up model, where the programmers have a goal to make software that fills a need in the market, and those same programmers implement that software.

These two models differ fundamentally in that they operate within different paradigms. Product-based engineering is suited for a smaller company or setup that has the goals of producing software that fills a need (often in the market). The project-based engineering operates in a paradigm where the product is not required to fulfill a market need / business idea: in other words, this model is more about fulfilling the needs of the customer. A real-world example where these differ is the scenario where a customer (boss) requests infrastructure software for the corporation (project-based): this software does not fill a business opportunity, but is requested by a customer, and therefore is project-based as opposed to product-based, as the end result is not a marketable business-idea-filling product, but is the result of a project that is company-specific.

QPM.2: Consider the largest software project you ever completed as part of a class assignment. How large was the project? How long did it take you to develop the system? Did it work perfectly, did it work most of the time, or did it fail the simplest test cases?

The largest contiguous project that I developed from the ground up in a class has been a web site, for CS361 (and a similar one for CS290, both of equal size and effort). These projects involved many html pages, database integration, client and server side JavaScript or Python, scripts to set up environment variables, scripts to populate databases. Further, the CS290 project covered a searchable database of tools (with images), and the CS361 project featured

API calls and stock prediction (non-AI). Each of these projects was completed in under a week, albeit with teams of 3 and 5 for projects in CS290 and CS361 respectively.

Initially, both failed some simple test cases. For example, the Python dependencies (we were using a Python Flask server to serve our web pages on our CS361 project) were different across each host system (we had teammates, customers and graders all using disparate operating systems and environments). This resulted in some versions of the software that failed to even start or install packages. Once remedied, the programs worked most of the time. Each failure was treated as an issue, so in the end, we had a site that worked quite reliably.

I learned something very useful when completing the CS361 project - our team forecasted to do a certain amount of tasks each week, but as finals week drew closer, most of my teammates were less and less able to complete tasks. This resulted in a project that fulfilled basic requirements, but that has many issues / tasks still outstanding. When thinking about how we could have remedied this, I will now consider (and ask about) other factors that could impact work ability among group members, such as finals, projects.

QPM.3: Identify and describe a business opportunity using the approach proposed by Geoffrey Moore. Then illustrate a product vision statement you would like to propose based on this opportunity.

I will fill out the Moore Vision Template for a product idea I came up with for CS361. I named it CFSD (Chimeric File System Device), and it is a USB key that is compatible with all operating systems by manner of serving data to a processor that can respond to the queries performed by the host computer in a manner consistent with the operating system's preferred FS on that machine. More info can be found here: https://github.com/lyellread/OSU_CS361/blob/master/homework/project-proposal.pdf.

Moore Business Opportunity: FOR all users of removable storage devices WHO need compatibility with many operating systems (or the option of compatibility with all operating systems) without erasing data on the drive during a format, THE CFSD (Chimeric File System Device) is a combination of existing hardware with new software in a USB-key form factor THAT provides the end user with the ability to reliably access the data on the CFSD from all systems based on that system's available FS drivers. UNLIKE standard USB Keys, which cannot be used across all systems without formatting or limitations (see link above), OUR product allows the user to connect and access data on the CFSD on any platform, regardless of OS and FS drivers.

Vision Statement: The CFSD will be a usb-key form-factored device that encompoasses a USB male port, a MCU and a variable sized (i.e. you can purchase different capacities of the CFSD, like a USB stick) Flash array. The software running on the MCU is designed such that it can make data requests (read, write) to the flash array. It will also be able to communicate with the host computer over USB, and will accomplish two tasks. First, it will be responsible for

determining or asking the host OS type. Second, it will interpret packets sent to it by the host, make the appropriate data access, and return the appropriate packet to the computer, in the format that the native FS expects. On a macOS machine, for example, when connected, the CFSD would discover that the host OS is macOS, and all communication between the CFSD and the computer will occur in a way that the drive appears to be formatted as APFS (macOS > 10.13) or macOS Extended (Journaled) (Macos <= 10.13) and function as a USB drive from there on out.

QPM.4: Explain the information source(s) you used for developing the product vision you have illustrated in the previous question.

Firstly, I had to assert that this is a problem: this is easy with personal experience and anecdotal experience shared by colleagues, friends. Next, I went about searching for solutions. This involved mapping out the issues with current FSs and finding how this table could be improved, or if there existed an FS already that could satisfy the requirements of cross-platform use (rendering the CFSD useless). The closest I found was the FS standard FAT32, but this has the limitation of only being able to accept 4GB files at max. Second to that is exFAT, which has the downfall of requiring drivers to be installed on most flavors of linux. I collected this information from the internet, as well as personal experience. Therefore, there is room for improvement.

Next, I considered the ways of implementing the CFSD, and because I was actively involved in some MCU / Embedded work at the time, that was the first thing that came to mind. It seems like a viable solution, so I decided to flesh that out. I determined that I would need (of course) a USB connection, and a flash array. Then, I also need at least one MCU with proper software to make the drive Chimeric, not a standard flash drive.

A couple issues included OS detection, which appears to be possible using Rubber Ducky like scripts (see link above). I would also need to keep this updated to know what the proper file system is for an operating system. To do that, I suggested a client that can be installed on a computer that can update (reflash) the MCU code to a newer version that includes updated FS to OS mappings and any bug fixes and improvements possible.