Software Engineering Homework 1

1) In your own words, describe why an engineering approach to software development is important.

Simply approaching software development as if only one individual will be working on the entire project is a bad idea. With increasing complexity, code becomes more difficult to revise and adjust for new changes in software. If the software development cycle did not contain any software engineering practices then all code would be messy, and personalized for only one developer. Software engineering is capable of choosing a method/process of approaching a problem. Because of this, software can have the benefit of being reliable and less expensive. With good practices, software developers can create readable code while tending to the most necessary needs of a piece of software. For example, if something was meant for dealing with software maintenance, then someone may want to choose an agile method so that they can keep up with any new bugs or defects. One of the biggest reasons to use software engineering practices is the ability to focus a team more on one goal instead of individuals having different ideas of what a product should be. For example, if a team was developing a game together, then they probably do not want every developer making his own features. This makes programs become clustered and lose focus of their original purpose. Software engineering is meant for teams of developers, and reflects heavily on the quality of the code in terms of readability, usefulness, reliability, and cost.

2a) For the following three software process models (also known as software development lifecycle – SDLC models) discuss each model's advantages and disadvantages in your own words.

- Waterfall – The idea of the waterfall model is to completely plan everything that will be done in the process of developing the software. By doing this, it will obviously be clear to the team what they need to accomplish each step and therefore can result in the program functioning as expected. In general, the waterfall is a heavily plan-driven process. Nothing happens with each step until the previous step is completed. This involves specification, design, unit testing, integration and operation and maintenance. One of the greatest benefits of this design philosophy is that is very cost effective; if they can plan out exactly how they will spend their money then they can manage it much better. However, the waterfall model does not work well for software that is not critical, hardware or something where the process obviously needs to be planned. Software often requires feedback from each cycle so that they can adjust the program accordingly. It is almost impossible to properly plan out everything that will be required during the development process. In this regard, the waterfall process fails to account for the limitations of a purely plan-driven practice.

- Incremental Development – This process is incredibly effective at producing software very quickly and cost effectively assuming that the requirements of the software is constantly changing. The issue with purely plan-driven models is that new requirements become costly, but with Incremental development these requirements can be interleaved into the development process. Another benefit of the incremental model is that it can use user stories to diagnose bugs and narrow down what the requirements are of the software quicker. Now, the software is not as expensive to change because it is literally built in expectation of the first idea of the product not anticipating all of the needs/requirements of a programs final version. There are two fundamental issues with this process, unfortunately. The first issue is that is not cost effective to visualize the process of everything in an incremental process. Managers will not be able to measure progress without this information, but its simply too expensive to make. The second issue is that the programs become significantly more complex as more increments are added. Readability takes a hit during this process, and it is often recommended to refactor the code to avoid this issue becoming too costly.
- Integration and Configuration – Probably the most cost effective of the 3 models, Integration and Configuration is inspired completely by the fact that there is already software out there that companies can use for a product. This avoids long and costly development cycles by simply using what is already there. To put it simply, it is much more preferable to make a program that uses assets already available because of speed of the development process and cost effectiveness. Sadly, it is difficult sometimes to meet the needs of users using only this approach. Integration and Configuration will limit a team to working with a specific framework, and sometimes this framework will not support all of their needs. So the biggest downside of this model is that it can possibly lead to not meeting all of the needs of a consumer because of limitations.

2.B.) Do you have a preferred model? Why or why not? Please explain.

I do not have a preferred model because having a universal model is not logical in some situations. If NASA used an agile incremental approach when developing the software for the rocket, this would require billions of dollars in testing, which just is not rational. It is obvious that when working with space travel today, waterfall is the best approach because it is designed to anticipate as much as it can, and then leading to a successful space launch. So it very much so depends on the situation on what model a team should follow. All three of these models are intended to be strongest in one area, Waterfall promises a well functioning product on its release, Integration and Configuration is very cost effective and can be done quickly, and Incremental adjusts well to new requirements. Because these models each have their own specific benefits and may work better in certain situations, it would

be ridiculous to have a preferred model. The choice of a model at the beginning of development is incredibly important because it will help significantly on the reliability and quality of the product.

3a)