**CS 347 Homework 1** 100 Points

Alex Johnson

1.6. One piece of software that the failure of could cause massive harm is self-driving cars. If something happened to the software of a self-driving car where it does not function properly could cause the car to crash which could be catastrophic if on a busy highway. Also if someone found a way to hack into multiple cars or whatever shared link between cars failed it could cause even more damage. This shared link could be information coming to the car through a server or Wi-Fi. Failure in this software could be catastrophic.

2.8. One way to combine processes is with the spiral process. This is a combination of the waterfall process and prototyping model. The team goes down the waterfall doing each step then tests it with the client. Then if needed, the team can go back to the top to repeat the steps. This can be used for things that need to be put into the market quickly like a game.

2.9. There are many advantages to developing software that is “good enough”. One of these examples is completeness, where all the requirements are met in the project. Another advantage is improved satisfaction from the customer because the project got sent out quickly. Finally, another advantage is efficiency. Not only is the product out faster but it is more cost effective because it took less time and resources to construct. One of the downsides to that is delivering a project in shorter intervals could cause errors that delay the final project. Also, a downside to this is with less time, there is less collaboration in teams and results in more individual commitment instead of teamwork.

3.2. Agility is the act of accomplishing things very quickly with full completeness. This also means being ready to accept any challenges presented to your team. Agile software means to be growth oriented and build on a project incrementally. Another piece of agility is keeping things cost effective. All in all, agility is building a project in the most cost effective and timely manner.

5.1. The first skill I believe is important is creativity. While some people have very good analytical skills it’s also good to be creative. Creativity allows you to think outside the box and find better solutions to issues. Technical skills are also important. Being able to be well versed in different topics of computer science can help you find different solutions or explain to people better and more efficient ways to do things. Finally, business skills are very important. Being able to communicate with the client well and work out what can be done and can’t be done is a key part in every process.

6.6. The most important principle out of the eight is to build an effective team. This one is the most important because it is the foundation for all the other principles. Without an effective team you cannot be agile, produce quality products, communicate, adapt, manage risk or create a valuable product. Software is only useful if it is built by a competent and strong team.

7.1. A lot of times, software developers may skip or glance over requirements in a project. This is because requirements are constantly changing for the same software. Typically, software developers will begin with a basic prototype and later looking to satisfy smaller requirements in the fine-tuning part. Some requirements may be skipped over if they are changed or not needed by the end of the project.

7.5a. Use Case: Withdrawal –

1. User prompts system to withdraw.

2. System displays the withdrawal UI.

3. User selects amount of money to withdraw.

4. System communicates with bank account to see if the money is there.

5. System gets response from account

6. If good system communicates with hardware

7. System prints the money

8. System prints the receipt

9. Systems checks to see if money has been taken.

10. Changes UI back to basic home page

8.1. It is possible to begin coding directly after the requirements are given. If we have the requirements, then the basis for the project can be constructed. However, there are some issues with starting immediately. The design of the project may suffer because the architectural design has not been considered. These interfaces may not be developed correctly, or well and global data structures may not be explicitly designed.

8.10. A sequence diagram depicts individual interactions taking place with the software. A prototype may have many difference sequences diagrams to analyze separate actions. One example of this is you may create a sequence diagram to withdraw money from an ATM. A state diagram however chats the entire system. This diagram explicitly states every single state the product can be in. For the ATM example it would show, deposit, withdraw, turn on, turn off and more. The similarity is each model depicts the communication between the actors and the product.