Kea Braekman

CMSI 401

10/30/19

Pilone & Miles Book Assignment

Problem 1 :

What are the two major concerns of any software project?

The two major concerns of any software project are the cost, and the length of the project.

Which do you feel is more important?

Speaking purely economically, any cost or sum of money appreciates with time. Therefore, we can draw a direct proportional relationship between cost and time. And to add a second layer of complexity, many developers are paid hourly, thus increasing the length of a project directly increases its cost. For these reasons, I would personally value time over cost.

Where does the idea of complete functionality fit with these two concerns?

Nearly every project requires both time and money. Complete functionality is an intermediary step towards finishing or releasing the project. This step ensures that all the features within the code is functional but still needs to be finalized due to bugs or stability issues.

Problem 2 :

In the Agile method of software development, what are the four main phases that occur in each and every iteration? Do you feel that any of them could be done at the start of the project and not

be repeated in every iteration? Do you feel that would save time overall on the project? Justify your answers with a brief explanation.

The four phases of the Agile method are : Planning, Implementing, Reviewing, and Releasing. No. I think all four of these phases are important for every iteration. The only phase I could imagine removing is the releasing phase. But this portion depends on the customer and the level of trust he/she has in the software development team. All three other phases I'd say are essential. Planning tasks is crucial, implementing is the most important part of the project and reviewing is essential. If the team would spend the most time on implementation, and the least on releasing, it would be beneficial.

Problem 3 :

In the Waterfall method for software development, what are the main phases that occur? How are they different from the phases in the Agile method? What other phases are in the Waterfall that are left out of Agile? Do you think these are needed in Waterfall? Describe a situation using Agile in which one of these "extra" Waterfall phases might me needed.

The waterfall method has the following phases : Requirements, System Design, Implementation, Integration, Testing, Deployment, and Maintenance. Waterfall differs from Agile because it doesn't follow Agile's sprints or iterations. The waterfall phases differ slightly when it comes to planning. Planning for Agile is similar to Requirements and system design for Waterfall. Implementation is identical for both methods. Reviewing for Agile is similar to Testing for Waterfall. Releasing for Agile is similar to deployment for waterfall, and Waterfall has the additional "maintenance" phase. The requirement and maintenance phases are left out of

Agile. I think that for a waterfall approach (without sprints), these steps are appropriate. It would be unnecessary to repeat at every iteration. If a team is working on a large project with quickly changing technologies, like web hosting etc… Requirements and especially maintenance can be extremely important.

Problem 4 :

What is a “user story”?

Natural story of a user that seeks a new capability or functionality.

What is “blueskying”?

Blueskying is brainstorming and gathering as many ideas before a project.

What are four things that user stories SHOULD do?

Describe one thing, use language that the customer understands, be written by the customer, and be short.

What are three things that user stories SHOULD NOT do?

Be a long essay, use technical terms that are unfamiliar, and mention specific technologies.

Problem 5 :

What is your opinion on the following statements, and why do you feel that way?

All assumptions are bad, and no assumption is a “good” assumption.

I disagree. Assumptions are like looking for something in the dark. It is mostly bad, and usually will not work. But every now and then, it can be done correctly. Some assumptions are correct/good.

A “big” user story estimate is a “bad” user story estimate.

I agree. User stories should be short. If a user story is too big, then it should be broken down into more user stories. These small user stories would be better.

Problem 6:

Fill in the blanks

You can dress me up as a use case for a formal occasion: **User Story**

The more of me there are, the clearer things become : **User Story**

I help you capture EVERYTHING: **Blueskying**

I help you get more from the customer: **Role playing, Observation**

In court, I’d be admissible as firsthand evidence: **Observation**

Some say I’m arrogant, but really, I’m just about confidence: **Estimate**

Everyone’s involved when it comes to me: **Blueskying**

I agree with the book answers.

Problem 7:

Explain what is meant by a “better than best-case” estimate.

A better than best case estimate is an estimate that is too optimistic and too short. All estimates should be as realistic as possible. And a better than best case estimate is not.

Problem 8:

In your opinion, when would be the best time to tell your customer that you will NOT be able to meet her delivery schedule? Why do you feel that is the best time? Do you think that would be a difficult conversation?

As early as possible. As soon as it is known to the team leader that the project is going to be late, it should be brought up to the customer. I feel that it is the best time because it gives time for the customer to know what to expect and to potentially adjust their demands. Yes, potentially. The customer has vouched for you, given you time and money, it can be difficult to confront them directly and tell them that the project is late.

Problem 9:

Discuss why you think branching in your software configuration is bad or good. Describe a scenario to support your opinion.

I think branching in my software configuration is mostly good. In some small scale projects that are more compartmentalized, it can be tedious. But on a large scale project like building a complex embedded website, (with databases, transactions, front and back ends…), one change can affect another previous change. Branching helps manage these discrepancies.

Problem 10:

Have you used a “build tool” in your development? Which tool have you used? What are its good points? What are its bad points?

I have not used a build tool in development.