Ria Singh, Anika Bhatnagar, Dev Raiyani, Gaurav Basu
Professor B.J. Johnson, PhD
CMSI 4071
October 24, 2024

Pilone and Miles Book Assignment

Problem 1

The two major concerns of any project are “how much will it cost?” and “How long will it take?”. We believe that between the two, cost management is more critical because exceeding budget can halt progress entirely. The idea of complete functionality fits between the time and cost of completing a project, as it directly influences both. Achieving complete functionality often increases the scope of work, which can extend the time required and raise costs.

Problem 2

In the Agile method of software development, the 5 main phases that occur in every iteration are brainstorm, design, development, quality assurance and development. Although most of the planning is done at the start of the project, I believe that all of the phases should be repeated in all iterations. Skipping any of the phases may save some time but risk further costs down the line in order to make possible adjustments.

Problem 3 - ani

In the Waterfall method, the main phases are Requirements Analysis, Product Design, System Design, Coding, Testing, and Maintenance, and they happen in order. In Agile, on the other hand, these phases are done in cycles, so the process is more flexible and adaptable with constant feedback. Waterfall also includes phases like Risk Assessment, Prototyping, and Exit Strategy, which Agile tends to skip because it’s more focused on short-term goals and adjustments. I think these extra phases are important for Waterfall because they help plan for the long term and manage risk in bigger, more rigid projects. Even in Agile, though, something like Risk Assessment could come in handy, especially in a project with lots of unknowns, like developing a healthcare app. In that case, identifying risks upfront might prevent bigger problems down the road, even if the project is using Agile’s flexible approach.

Problem 4

Write one-sentence answers to the following questions:

What is a user story? : A story of how users interact with software that you’re building.
What is blueskying? : While brainstorming, think of every possibility/idea while focusing on the core software requirements.
What are four things that user stories SHOULD do?

- Describe one thing that the software needs to do for the customer
- Be written using language that the customer understands
- Be written by the customer

- Be short, aiming for 3 sentences

What are three things that user stories SHOULD NOT do?

- Be a long essay
- Use technical terms that are unfamiliar to the customer
- Mention specific technologies

Does the Waterfall method have user stories?

The Waterfall method typically does not use user stories in the same way that Agile methodologies do. In Waterfall, the development process is linear and structured, with distinct phases. Meanwhile, Agile incorporates user stories very commonly and emphasizes user perspectives throughout the duration of the project.

Problem 5 -

Problem 6 -

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: Observation
I help you capture EVERYTHING: Blueskying
I help you get more from the customer: Role playing
In court, I'd be admissible as firsthand evidence: Observation
Some people say I'm arrogant, but really I'm just about confidence: Estimate
Everyone's involved when it comes to me: Planning poker

Problem 7 -

Explain what is meant by a ‘better than best-case’ estimate.

A "better than best-case estimate" refers to a projection that surpasses even the most optimistic scenarios. In project planning, best-case estimates usually outline the most favorable outcomes based on ideal conditions, such as all tasks being completed on time and without issues.

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? If so, how could you make it less difficult?

In my opinion, the best time to inform a customer that you will not be able to meet their delivery schedule is as soon as it is reasoned that there are no alternatives to meeting the deadline. This timing is ideal because it demonstrates transparency and respect for the customer's needs, helping to maintain trust. If you wait too long, it could reduce customer satisfaction and lead to increased frustration.

I think this will be a difficult conversation because no client would ever want to hear about delays to their current project. Increased time delays could lead to additional costs, increased

work time, and a delay to completing further tasks. To make the situation less challenging, it is important to first make sure they are fully informed of the scope of the problem to communicate to their company. To do this, gather all relevant information about the reasons for the delay and potential impacts. To mitigate some of the frustration, present alternatives, such as a revised timeline or potential ways to eliminate the delay. Finally, check in with the customer to provide updates and show that you're actively working on the issue.

Problem 9 -

I think branching in software development is a good practice, especially when working in teams or on complex projects. It allows developers to work on new features or bug fixes without disrupting the main codebase, which helps keep things organized. However, it can become a problem if it's not managed well, like when branches stay isolated for too long, leading to merge conflicts.

While working on the Momento project, I found that branching is generally useful, but it can also introduce small problems if not handled properly. For example, if I create a new branch to tweak the UI for the photo capture feature and someone else is updating the navigation layout at the same time, I might end up with a conflict when I try to merge my changes. It's a minor issue, but it can disrupt the workflow.

If we both had been frequently pulling updates from the main branch and merging regularly, the conflicts would be much smaller and easier to resolve. This shows how even small things, like delaying merges, can cause unnecessary problems.

Problem 10 -

While working on Momento, I have used some build tools to streamline the development process. One tool I frequently use is **Xcode**'s built-in build system for iOS development.

Good Points:

- **Seamless Integration**: Since Xcode is the primary IDE for iOS development, its build tool is tightly integrated into the workflow. It handles everything from compiling code to deploying the app on a simulator or a real device, making it very convenient.
- **Automatic Dependency Management**: Xcode’s build tool manages dependencies and frameworks, such as **CocoaPods**, automatically, ensuring everything is linked properly.
- **Error Handling**: It provides clear and detailed error messages during the build process, helping to quickly identify and fix issues.

Bad Points:

- **Slow Build Times**: Sometimes, especially with larger projects or when multiple frameworks are involved, build times can be slow, which can be frustrating during frequent testing cycles.
- **Complex Configurations**: Managing build settings, particularly for different environments (debug vs. release), can get complicated and overwhelming in Xcode's interface, leading to errors if not carefully managed.

Overall, I like how Xcode's build tool is fully integrated into the iOS ecosystem, but I sometimes wish it were faster and easier to configure for complex projects.