Pages 280-281:

The Real Estimation Process:

Question 1:

I know there's merit to what this developer says, however I know that he is being far more pessimistic than he has to be. Trying to estimate times for projects is very difficult, as I have learned from talking to other senior developers, as well as from my own time working on projects at Koch Industries, estimating time is difficult, but not impossible. Our own huge project we were working on at Koch wasn't a death sentence either when our project went both overtime and over budget, but it was both of those, and considering the scope of the project, it wasn't a surprise. However, I have also seen projects go on time, and it was thanks to a lot of good experience working in software development, correct staffing, a hardworking team, and lots of consideration when it came to errors in the process. Truly, a good project estimates a project will take at least 10% longer than you think you should thanks to breaks, emergencies, staffing changes, training, and more. When you put these together, you get longer then what you should. As well, the business always wants things to be done faster, but depending on your employer and your managers, your project may get the time it truly deserves.

Question 2:

I can see how some managers would do this. There is an idea called the Triple Constraint when it comes to project development, where you have three factors: time, resources, and cost. If you make one longer, the other two have to compensate and vice versa. For instance, if you decrease time, the thought is that cost and resources can go up in order to compensate. However, while the triple constraint is great in theory, there is some thought applied that a project cannot always be

completed faster when more people are added to it. My favorite quote involving this is this one, “9 women can’t make a baby in a month”. Sometimes, the nature of a job requires time. In my opinion, I do believe that managers should work with IT more and negotiate less, but keep IT to their word if they go along with their time schedule instead. As well, the business should be smart and adapt to technological changes before they are forced to make them. No one should be working 60 hour weeks without a choice.

Question 3:

I think it costs more having a schedule of 11 months with a 1-month overrun, simply because the business will think of ways to make sure everything runs to completion by adding resources that the project may not have needed. By overestimating as well, you may be able to add more features in a positive manner, once the main work is done, which is a positive effect of Parkinson’s law.

Question 4:

I would simply allocate more time for these things. The future is unknowable, and punishing others for the unknowable is not something I like to practice. Plus, I like to be realistic in my estimations instead of optimistic.

Question 5:

This is dangerous behavior because it leads a team to believe that whenever they are given a time estimate, they will have to go over it. Projects, especially big ones, are known to go over, but there are short term projects that can be completed reasonably. As well, it conjures a negative image of the business vs. the information technology staff that works for the business, and can

create hostile environments between the two groups of people, which is not good for team building.

Question 6:

I would show him some examples from my past portfolio of projects where the scheduling was correct, or even overestimated. I would also ask him to try different methods I've learned to try to estimate times, so we can get as close as possible to an accurate time. Just because we're wrong, doesn't mean we can't shoot for perfection. If he was stuck in his ways, I would look at action against him, only after much discussion with him about changing his attitude. Culture is far more important than skill in my book.

Question 7:

I think it stems mostly from the fact that not everyone knows how to code. If business managers did understand how exactly coding worked, and how to develop applications to make them usable, there would be better time estimates. I also think it's due to the rapidly changing nature of technology. Every year there's a new popular language to develop in, a new framework, or a new architecture that lets applications do incredible things, like run in Docker instances so each environment can be treated the same. The fast-paced nature of how our landscape works means that the technology that built an application five years ago is most likely outdated and needs to be changed almost completely to keep up to date with security and architecture. There is merit to using older languages and development methods, however you must be careful when tying them to a company or a certain software, like Microsoft or Apple. I know successful developers still using PHP since it is open source, but many companies struggling since development behind Visual Basic is dying. As for how projects are the same, they still involve the business, still

involve people, and good people management skills, as well as tasking a team to work together and think of the best solutions for the most people at a workplace. Besides the tools we are using, and the basic understanding of what we are building, there are few differences.

Question 8:

First off, I really think this will be less of an issue as time goes on. More schools are teaching software development skills, and knowledge behind how code works. Therefore, I think in the future, more people will know how to code. As for other things managers should do, I would also ask when estimating how long a project will take what technology they think they will use. Instead of using the brainstorming time within a project to see what language and what platform an application or project should be using, that should be asked when developing the project charter, so that people the business can ask real questions and learn more about what the advantages of newer frameworks are to see if that is something they should invest more time into, or if they should stay with the same technology.