Chapter 3 Homework

Tanis Olesen

> 3.3. Extreme programming expresses user requirements as stories, with each story written on a card. Discuss the advantages and disadvantages of this approach to requirements description.

- Requirements can be formed based on how the customer describes the tasks. You can get an insight on how the customer might use your system. User Stories can often miss details, requiring further communication with the customer to clarify requirements.

> 3.4. In test-first development, tests are written before the code. Explain how the test suite may compromise the quality of the software system being developed.

- Tests ensure that the code you write accomplishes a task correctly. If there are too many test cases, it could take an inordinate amount of time to run tests. Taking longer to ensure that the code you write does not effect the rest of the system. This could slow down the development process, with programmers twiddling there thumbs waiting for tests to finish. Especially if the testing suite is bulky.

> 3.5. Suggest four reasons why the productivity rate of programmers working as a pair might be more than half that of two programmers working individually.

1. More time is spent communicating between the programmers than the code getting written.
2. Ideas take longer to get fleshed out as another person has to understand and vet the idea.
3. One person could slow down the other because idea's have to constantly be clarified.
4. When writing code, tasks have to be split up between the two, and consistency of logic must be maintained.

> 3.8. Why is it necessary to introduce some methods and documentation from plan-based approaches when scaling agile methods to larger projects that are developed by distributed development teams?

- To maintain integrity of the architecture. Ensure tasks are getting distributed evenly. Maintain consistent formatting and readability of code. Make sure all requirements are getting met.

> Pick 6 of the 12 principles behind "the agile manifesto" that mean the most to you, and describe to me why they are valuable in the software process.

1. > The most efficient and effective method of conveying information to and within a development team is face-to-face conversation.

   - I believe this is even truer after covid. Even thought technology may be advancing to bridge the gap between the digital world and the real one. Communicating in person allows for vast amounts of non-verbal communication to happen. Allowing for better conversations to happen.

2. > Build projects around motivated individuals. Give them the environment and support they need, and trust them to get the job done.

   - It is important to let people have an environment and support that allows them to succeed. When you have motivated individuals you don't have to worry about them being lazy about requirements.

3. > Business people and developers must work together daily throughout the project.

    - This allows for ideas to be passed around between departments easier. As the departments are not compartmentalized. A better understanding of a project from all individuals allows for work to get done more efficiently.

4. > Welcome changing requirements, even late in development. Agile processes harness change for the customer's competitive advantage.

    - This gives software the freedom to evolve to the customers needs. Getting the software solution to meet every requirement the customers need, even if those requirements were introduced late.

5. > At regular intervals, the team reflects on how to become more effective, then tunes and adjusts its behavior accordingly.

    - An organized, well communicating team. Can overcome difficulties with ease. As the teams process evolve and get better and more efficient the team can work faster. Delivering working code to the customer sooner.

6. > Working software is the primary measure of progress.

    - By basing progress by working software, it instantly shows you what still needs to be worked on. Giving a clear path that progress is not completed without a working solution.

> List 3 additional questions you have about Chapter 3 or agile.

1. Would you want a team to always be working the same hours?
2. In what other ways can Test Driven Development degrade quality of product?
3. How do you recover from problems that prevent progress from being made?