Jasir Massey Campbell

Software Engineering

Assignment 3

**4.1.** Since a focus on quality demands resources and time, is it possible to be agile and still maintain a quality focus?

The agileness of a project can vary depending on the size of the project. This means that the scope used to determine the agileness of and project will vary in size as well. So, although a focus on quality requires time to be most efficient, the goal of software engineers should always be to have an agile approach. An agile approach is a focus on conciseness. This approach will provide quality software and should always be maintained regardless of size of the project or type of project.

**4.2.** Of the eight core principles that guide process (discussed in Section 4.2.1), which do you believe is most important?

All of the principles can be argued as being most important as they are all very important in the scope of producing quality software. If I must choose, the most important of them all is principle eight. The entire industry of being an engineering rests on being able to provide a more effective way of doing something for the user. Without the use there is no financial benefits in creating software, which means that there would be no means to pay engineers, which would mean a stalemate to the progression of the industry. The other seven principles focus more on the software itself, whereas with principle eight, it acknowledges that the software only matters if it is providing value to the user.

**4.6.** How does agile communication differ from traditional software engineering communication? How is it similar?

Agile communication is similar to more traditional styles that slow the process down by both describing the communication between the parties relevant to the software’s development. They both have an end goal of getting software to the end user and they both want to achieve creating quality software. The difference between the two styles is the approach taken to get to those end goals. Agility focuses on getting software to user as fast as possible. Nevertheless, this doesn’t mean that it doesn’t account for quality as well. With tradition methods, quality is achieved through detailing and elongating the development time in effort to avoid mistakes. With agile methods, quality is managed by not over-doing things and focusing on simplicity and conciseness. Creating too many models or being over-detailed can lead to creating a situation that is less flexible and adaptable.

**4.10.** Why are models important in software engineering work? Are they always necessary? Are there qualifiers to your answer about necessity?

Models are important because they help keep the teams focused on a central goal and stay organized and it simplifies the communication process between the various parties involved. Are they necessary? Not necessarily. Im sure that experienced programmers are able to create programs during solo projects where models to them are unnecessary. In the same token, I would also assume that experienced programmers would prefer having models and structure when working on large projects that involve factors such as the technical literacy of a co-worker or stakeholder or the skill of the other programmers involved. Having a model would making managing these factors much easier.

**4.13.** What is a successful test?

According to the textbook, a successful test is one that uncovers an as-yet-undiscovered error. The purpose of testing code is to avoid bugs that can prevent the software from running at all or as intended. A successful test is one that has the ability to discover any error in the code that would prevent it from running as intended by the engineer.

**4.14.** Do you agree or disagree with the following statement: “Since we deliver multiple increments to the customer, why should we be concerned about quality in the early increments—we can fix problems in later iterations.” Explain your answer.

I think that the statement above is completely unprofessional. The intent of engineers should always be to present the user with a quality product regardless of which increment it is. Having the mindset that a particular problem can be addressed later is naive because it does not account for the likely problems that will arrive in the future.

**4.15.** Why is feedback important to the software team?

Feedback is a critical part of the development and improvement of the software industry and definitely for a software team. Debugging allows the engineer to find discrepancies in the code that can prevent it from running on a programmatic level. It cannot detect other problems that might arise on a user interface level. For example, specific details such as which software should be created/designed, the preferred size of a device or preferred color of a UI is valuable information that comes from communication with the user base. Bugs or other complications in more complicated software involving AI may only be found after the launch of the product. For example, video games often require a major patch shortly after release to fix bugs that could only have been found through the thousands to millions of different occurrences performed by the user base. This is especially true in mass multiplayer online games.

**5.1.** Why is it that many software developers don’t pay enough attention to requirements engineering? Are there ever circumstances where you can skip it?

Developers can sometimes undervalue requirements engineering due to many factors including being overzealous, over-confident, lazy or more. Nevertheless it is an important bridge in determining roles and developing plans of action. I think that requirements engineering should never be skipped on serious projects. Projects that allow the programmer to be more flexible or projects that beforehand are recognized as going have extreme short term changes may consider skipping the requirements modeling phase.

**5.4.** Why do we say that the requirements model represents a snapshot of a system in time?

Because it is ever changing. The model should be expected to continuously be dynamically changing.

**5.6.** Develop at least three additional “context-free questions” that you might ask a stakeholder during inception.

1. Who is the audience the software is for?
2. What problem of their’s are we solving?
3. What is the timeframe to deployment?

**5.10.** What do use case “exceptions” represent?

A use case exception describes an event that does not lead to the use cases goal. The most common of these is during user data entry when a user may have to cancel the use case. Leading the the use case goal never being met.

**5.11.** Describe what an *analysis pattern* is in your own words.

An analysis pattern is basically a recognized or discovered reoccurring issue . The goal is to quicken the requirements phase by providing reusable analyzations.

**5.12.** Using the template presented in Section 5.5.2, suggest one or more analysis pattern for the following application domains:

a. Accounting software : arithmetic functions (add, sub, etc..)

b. E-mail software : create new email, receive email   
c. Internet browsers : take in url, display url data  
d. Word-processing software : display output

e. Website creation software : pallet box (quickly add shapes, changes fonts, add pictures)

|  |
| --- |
| 6.3, 6.4, 6.5 |

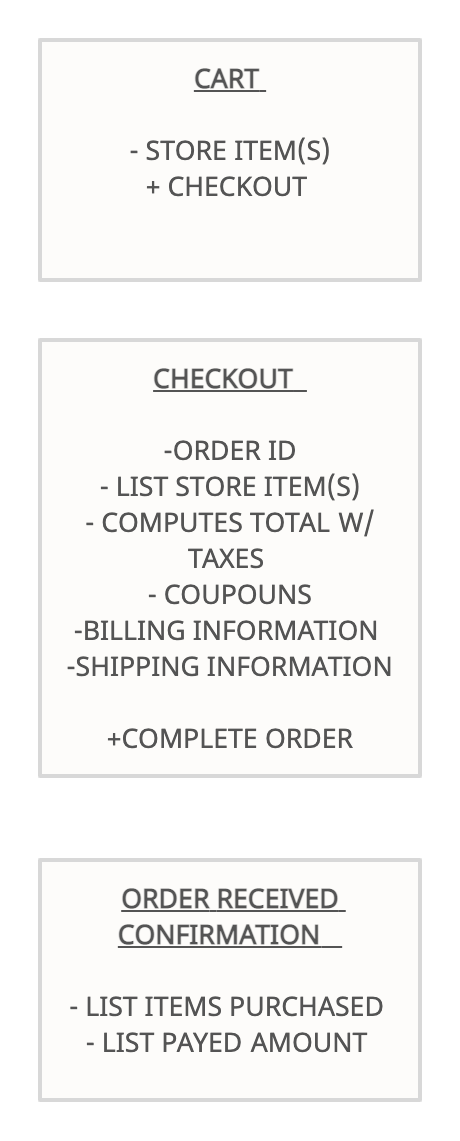
**6.3.** What is the purpose of domain analysis? How is it related to the concept of requirements patterns?

Domain analysis promotes reusing software by the creation of analysis patterns and enables engineers to analyze and solve problems faster, quickening the development cycle. During domain analysis, requirements are specified to satisfy the solution domain.

**6.4.** Is it possible to develop an effective analysis model without developing all four elements shown in Figure 6.3? Explain.

The purpose of the analysis model should be to acknowledge the all of the properties as concisely as possible. So it is possible to develop an effective model with some of the elements outweighing the others. The drop-off is that unless is it specifically done to accommodate a specific need, a more complete analysis will include all elements.

**6.5.** You have been asked to build one of the following systems:

This order processing system labels the 3 web pages/items, the contents of pages and their relationships to one another. Cart > Checkout > Order Confirmed 

**7.1.** What is the fundamental difference between the structured analysis and object-oriented strategies for requirements analysis?

The main difference between structured analysis and object oriented analysis is that structured analysis has a larger focus on the procedures of a system (in hopes to understand the system in a logical way) whereas, with object oriented analysis, the focus is more on the relationships between data structures/ or objects.

**7.8.** How does a sequence diagram differ from a state diagram. How are they similar?

Sequence diagrams are meant to describe a sequence within a system. It describes the execution of a use case and the path or paths to the use case’s goal. A state diagram has the capability of describing all of the functions within a system or a single method. It shows what functions are valid to the project.