Homework 08, CPSC-4175

Chapter 3, Object-Oriented and Classical Software Engineering

Octber 9, 2017

Note: The text for this class uses some terms in ways that are not consistent with the way that I have used these terms. Software engineering has not been systematized to the point where all terms have one meaning, and only one meaning. To some extent this is true throughout computer science, for example, "variable" sometimes means a single assignment variable (i.e., a constant), sometimes means only that which can appear on the left hand side of an assignment operator, and sometimes means an expression or a token that can appear on both the right hand side and left hand side of an assignment statement. The key is to understand the *concepts* to which the term refers, and not to get confused by the fact that sometimes the same word can refer to different concepts. When you work as a software developer, you will learn to adapt to the terminology of your employer, even though it may not conform to your previous experience. Again, don't get confused when the words don't mean what you might expect them to mean.

1. The book notes that the *Unified Process* is not "a specific series of steps." It notes that "[i]nstead, the Unified Process should be viewed as an adaptable methodology." Given this, how should you approach the content in this chapter that seems to present the Unified Process as a specific series of steps? How would you approach this chapter if you in fact viewed the Unified Process, not as a process, but simply as a methodology?

2. The book states that "[t]he object-oriented paradigm is an iterative-and-incremental methodology." Do you think that this is a true statement? How is this similar to our class discussions of a spiral approach to software engineering? Is there any difference between these two views, and if so, what is the difference?

3. In class, we are using an iterative process with four phases: analysis, design, implementation, and testing. The book discusses the *five core workflows* of the Unified Process. List the five core workflows, and for each, give a *brief* summary of the workflow.

4. What is the relationship between the *software project management plan* and the software development process? Do you think that a software project management plan can replace an explicit software development process? Why or why not?

5. Section 3.7 discusses testing with regard to four process areas: requirements artifacts, analysis artifacts, design artifacts, and implementation artifacts. What is the difference between the test workflow discussed in the chapter, and the testing of the artifacts produced in each of the five process areas? Do you think that this is an inconsistency in the book's treatment of testing?

6. If it is true, as the book claims, that "[p]ostdelivery maintenance is not an activity grudgingly carried out after the product has been delivered and installed on the client's computer," how do you fit postdelivery maintenance into an iterative, spiral development model?

7. Look at figure 3.10 on page 88 of the book. This figure seems to suggest a model of five workflows (requirements, analysis, design, implementation, and testing), with each workflow consisting of four phases (inception, elaboration, construction, and transition). In class, we have talked about an iterative, spiral model with four phases for each iteration (analysis, design, implementation, and testing). Do you think that these two approaches are mutually exclusive? Do you think that they are mutually supportive? Before you answer, review your response to the first question.

8. Look at figure 3.2 on page 94. In a sense, the waterfall method can be easily adopted to an iterative, spiral development model, using the waterfall approach to model each iteration, i.e., not a “waterfall” but a series of rapids, or a cascade. Viewed in this way, how do you think model 3.2a and 3.2b would actually differ in a real-life project? Do you think that they would basically amount to the same thing, or do you think that they would be fundamentally different?

9. The last section discusses *software process iprovement*. What is the difference between software improvement, and software process improvement? How important do you think that software process improvement is in the field of software engineering?