Jasir Massey Campbell

Assignment 4

Software Engineering

**8.1.** Do you design software when you “write” a program? What makes software design differ- ent from coding?

Prior to just a few years ago, after years of having been in college, I did not have a good understanding of the difference between software engineering and programming. Even after taking years of high-level programming courses and other computer-centered courses I did not understand the difference between the two as well as I do now. Programming is a term that basically solely encompasses the writing of computer scripts. Software Engineering includes the entire process of getting the product developed and to the user. Programming is just one of the activities that Software Engineering includes. Software Engineers can be responsible for managing multiple programming teams in conjunction with spearheading the dialogue between the technical employees and stakeholders.

**8.2.** If a software design is not a program (and it isn’t), then what is it?

Software design includes the entire process of “designing” the software. Meaning, that a software designer has the main responsibility of bringing the project together. A designers responsibilities could includes planning, communicating, designing models, hiring, and more. Whereas a programmer may only be responsible for some of those same things, but on a much smaller scale. The main expectation of a programmer is to code not to manage.

**8.3.** How do we assess the quality of a software design?

Software design is important to engineers because it allows them to predict the quality of software before it is ever created. According to the textbook, some features that are used to assess software design quality include, functionality, usability, reliability, performance, and supportability. Although briefly mentioned, the textbook does mention the use of Technical Reviews (TRs) which are meetings consisting of programmers carrying out roles with the end goal of assessing software quality.

**8.4.** Examine the task set presented for design. Where is quality assessed within the task set? How is this accomplished? How are the quality attributes discussed in Section 8.2.1 achieved?

One can argue that virtually every step has an aspect of quality design to assess some part of the software design process.

* In step one there is in emphasis on quality assessment by the specification of designing “appropriate” data structures.
* In step three I would like to highlight the task instructing the engineer to “be certain”
* In task four, it instructs to “check each design class against design criteria” and to “evaluate and select design patterns”.
* Task six instructs to “review results of task analysis” and to “review the interface design and revise as required”
* Lastly, task seven requires the engineer to “refine the interface of each component.” and to review each component and correct all errors”.

Most of the task of sub task that feature some type of quality assessment. Language that hints to quality assessment is put in the “ “ above. But the step that focuses most on quality assessment is that last step that goes over the entire project to review each component.

**8.6.** Describe software architecture in your own words.

Software architecture describes the relationships between the major elements of the software. According to Shaw and Garlan (the textbook), structural property, functional property and related systems should be described within the software architecture.

**8.8.** Describe separation of concerns in your own words. Is there a case when a divide-and- conquer strategy may not be appropriate? How might such a case affect the argument for modularity?

Separation of concerns is an idea based around the belief that any complex problem can be easier and more manageable to handle if it broken down into smaller pieces. It is rare for modularity to not benefit a software team during development. The only time that modularity can become a problem is when the there is so much modularity that the software mangers become overextended and can’t keep up with the subdivision. But this is concern revolves more around ineptitude than a lack of integrity with the idea of divide and conquer.

**8.10.** Discuss the relationship between the concept of information hiding as an attribute of effective modularity and the concept of module independence.

One of the great benefits of information hiding is in the simplicity of debugging. The textbook states that “inadvertent errors introduced during modification are less likely to propagate to other locations within the software.” In essence, what is being stated is that it is hard for an engineer to mess up something that they do not have access to. Effective modularity of software refers to the software being composed of independent modules. Module independence is assessed through the examination of software by using the techniques of coupling and cohesion.