Anthony Menjivar

CMSI 402

Prof. Johnson

February 4, 2015

Assignment 1

**1.9 - When implementing a control program for a dishwasher that has fewer than 10 different washing cycles, what paradigm would you use? Why?**

Here I would use the waterfall paradigm. Looking at the model of software lifespan through the waterfall approach, we need to be given and have ready any and all requirements for the dishwasher program as well as the full design of it. Once we are given those two things then we can start the implementation of the program. Lastly, once the software is implemented, it ends up in the last stage of the waterfall which is the maintenance stage after being given to the user. This works for a dishwasher program because all requirements and design can be given beforehand and with no need to be updating any software, the program can stay in the maintenance phase once being given out to users. This goes along with the fact that in the waterfall paradigm, there is no flowing backwards within the lifespan of the program.

**1.10 - When implementing a website with 10 different pages for a medium-size business, what paradigm would you use? Why?**

In this case, the waterfall method would not work very well because of the volatility of requirements that implementing and maintaining the website. Therefore, in this case we would need to use the iterative approach in order to be a ready for any software changes or updates. Maintaining a website has many requirements but on top of that, these requirements at times change due to any new software that may be released that will affect the website, or new and better ways of implementing the website, and even just the business possibly saying that they want things changed. Iterative works here because the website is being prepared for a limited time as it can be changed or updated at any other time.

**2.8 - Why is the term life cycle misleading? Which term is more commonly used: life span model or life cycle?**

Life Span is the term that is more commonly used. This term is misleading because there really is no cycle when it comes to the “life cycles” of software. It may be assumed that when one program ends, another program surfaces to take its place but that is not true as when most programs come to an end, there is no other program that will replace it.

**2.9 - In what situation can the V-model of the software life span be used?**

The V-Model of software life span should be used in the situation that there is a large emphasis on testing. This is because it’s starts off by following the waterfall model with the design being split into system and unit design but once the software is implemented, the model requires that developers go back to each an test each thing before putting it into the maintenance stage.

**2.10 - What are the advantages and disadvantages of the prototyping model?**

The advantages of the prototyping model is that it is fast and tentative implementation that focuses on the verification of the requirements and presents them to the user as a provisional first version. This helps because a lot of the time when users see software for the first time, they say that they wanted something different. Although this helps with part of the volatility problem, the disadvantage is that there is still volatility that comes out after the implementation.

**3.13 - What is inheritance in object-oriented technology? Give an example.**

Inheritance is a relation between two classes, the base class or superclass and the derived class or subclass. The super class defines the base type and class members that are shared among all subclasses while the subclass defines the derived type and contains members specific to that super class. An example would be a super class called *Person* with certain attributes such as *name* and *address* while a subclass called *Customer* implements the *Person* class and inherits the *name* and *address* while also implementing new attributes such as *account*, *deposit*, and *withdraw*.

**3.14 - What is the difference between an object and a class on OO technology?**

In software, objects are what represent the objects of real world. These objects then have properties of their own for example: An account has a balance, and customer has a name. These objects also perform certain actions (ex. A customer withdrawing money). A class in software is the module that defines the properties of the objects. The class also contains functions and are sometimes called class methods.

**3.15 - Describe the role of polymorphism in object-oriented technology. Give an example.**

The role of polymorphism in object-oriented technology is to serve as a sort of switch statement for the class. Base types appear in the code but the switch makes sure which derived object is being invoked. For example making classes of different farm animals: *Cow*, *Sheep*. These classes inherit a method from a class, *FarmAnimal*, called *makeSound()*. When calling the *makeSound()* method in each class, there will be a different output because of the class outputs, making it going along with the fact that a cow and a sheep make different sounds

**3.16 - Describe the role of "information hiding" in program comprehension.**

The role of “information hiding” has to do with objects and making the complicated and messy details of the object. While doing this, it also presents the object as a clean and simple interface.

**4.1 - Draw a class diagram of a small banking system showing the associations between three classes: the bank, the customer, and the account.**

Customer

Account

Bank

TakeAction

BankRecords

Balance

CheckBalance

Withdraw

Deposit

**4.9 - Explain the meaning of the activity diagram in Figure 4.15**

The activity diagram starts at a checkout. Next it goes to an edit changes activity followed by a decision in the activity diagram. The decision is whether you are working with a team. If the answer to that decision is not then the next activity would be the commit and then ending the activity diagram. But if the answer to the previous decision were to be yes, then we go to an update activity which then brings us to another decision. The decision is whether there is a conflict or not. I no conflict, then the next activity is commit and then ending the activity diagram. If there is a conflict, then we first hit the resolve conflict activity, then hit the commit activity, until finally committing and ending the activity diagram.