**SE 361 – Practice Exam #2 Jenny Zhen; 10.21.12**

1. Usability is very important in the design of a user interface. List and describe four usability factors to consider when designing a user interface.

* Consistency
* Readability
  + Good color scheme
* Relevancy of motion
* Navigability
* Ease of use
* **Communication**
* **Controlled autonomy**
* **Efficiency**
* **Focus**
* **Fitt’s Law**
  + **The big buttons are easy to click**
* **Learnability**

1. What are three user interface design models?

* **User Model** – profile of the end users of the system
  + Who is using the system? What they think when they are using the system?
* **Design Model** – profile of how the system is designed/intended to work
* **Mental Model** – how the user thinks the system works
* **Implementation Model** – how the system actually works
* **The design model tries to make all the other models work.**
* **The goal is to make the design, mental, and implementation models nearly the same.**

1. What is an anti-pattern?

* Superstition; something a lot of people do that isn’t a good idea
* **Using something that seems like a solution, but does not fix the problem**
* Using the design pattern in an inappropriate context, or incorrectly

1. List three reasons why unit testing is a good thing to do during software development.

* Finding bugs early on (cheapest to fix early on)
* Rerun tests; find regressions (catch errors in other parts of the code)
* Prepare for integration testing
* Less defects
* Easily automated

1. In addition to the “solution” section that actually describes the pattern, the documentation for a design pattern includes “context,” “problem,” and “forces” sections. What is the purpose of each of these three sections?

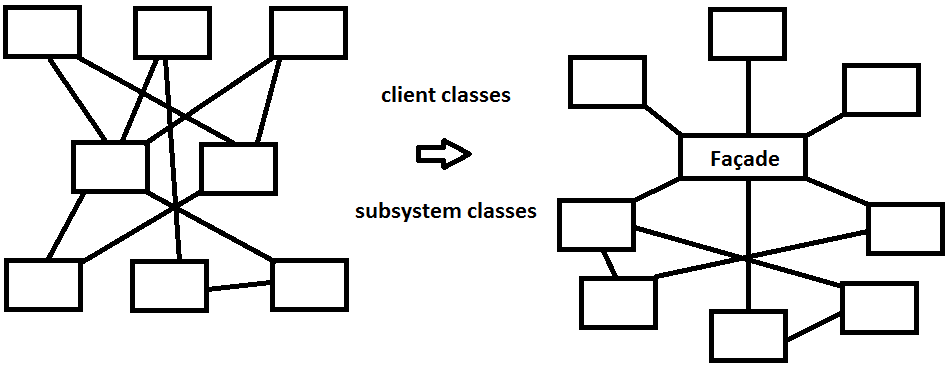
* **Context** 
  + Where the problem exists and factors to consider
  + The general situation where the problem occurs; possible domain
* **Problem**
  + A brief description of the main difficulty the pattern is trying to solve
* **Forces**
  + Issues or concerns to consider when solving the problem

1. What is the purpose of the Singleton pattern? Describe one of the forces this pattern encounters.

* **Ensure that it is never possible to create more than one instance of the class**
* **There can never be more than one**
* **Provides global access to the one instance; everyone has access**
* Used only when everyone needs access to the one instance
* **Forces (requirements)**
  + Public constructors cannot guarantee that only one instance is created
  + Singleton instance must be accessible to all classes that require it

1. Another co-op student is working on the same project that you are working on. He shows you some code in which he says he used the Façade design pattern to isolate a subsystem from the rest of the program. You look at the code and see that each function in the Façade calls a single function within the subsystem. You tell him that is not a good example of a Façade. Give your answer below when he asks what is wrong with his Façade and how he should fix it.

* **The objective of Façade is to simplify the interface and reduce the number of method calls necessary by the client**
* **Should be making the task simpler**
* Cover up something to make it look prettier



1. What is the term for unexpected behavior exhibited by a system when observed by the user?

* **Defect/Fault**– problem in the system found by the end user after software is released
* **Error** – problem found by the QA (quality assurance) or tester team
* **Bug** – problem found by the developer

1. Identify and describe four defined roles in the formal inspection process.

* **Scribe/Recorder** – records any problems
* **Reader** – interprets the code for the rest of the team
* **Moderator** – helps to keep the discussion on track; follow-up (makes sure that problems have been fixed)
* **Inspectors** – go through and prepare comments before the meeting; provides the input and contributes to discussions during the meeting
* **Producer/Author** – distributes the code ahead of time; arranges time for the meeting; usually silent during the meeting, unless if there is a large discrepancy

1. Identify two advantages of using a formal inspection process over informally looking over each other’s code.

* Written record, accountability
* Confirmation of defects
* Everyone is informed about the defect
* Everyone is prepared
* Refutations/comments are facilitated (keeps people happy)

1. What advantages do formal inspections provide over unit testing? What advantages does unit testing provide over formal inspections?

* **Advantages of Formal Inspections**
  + Verify logic issues
  + Find defects in maintainability/efficiency
* **Advantages of Unit Testing**
  + Faster testing; faster fixing
  + Does not require other people
  + Actually running the code
  + Makes sure it works correctly in the first place
  + Re-runnable

1. List two types of coverage that is applicable for unit testing.

* Coverage – ensuring that every part of the system has been tested
* **Functional Coverage** – ensuring every function has been tested
* **Statement Coverage** – ensuring that every statement has been covered
* **Path Coverage** – ensuring that the ideal path has been covered
* **Branch Coverage** – every individual branch/path of execution has been tested (if/else if)

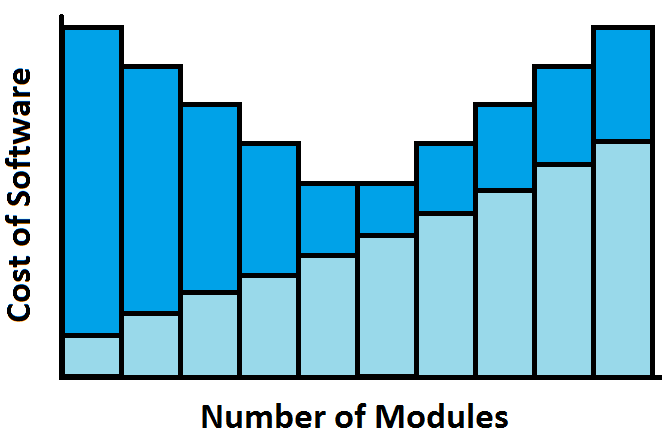
1. Describe three types of testing performed by users and clients.

* In chronological order:
  + **Usability Testing** – testing with the end users/clients
  + **Alpha Testing** – bring clients in to work with you (and you creep over their shoulder as they work)
  + **Beta Testing** – you give clients the software, as if it was the final release, and you let the clients use the software; problem are reported back
  + **Acceptance Testing –** what happens at the end

1. What type of testing is performed when changes are made to a system?

* **Regression Testing** – re-running test cases
* **Smoke testing**

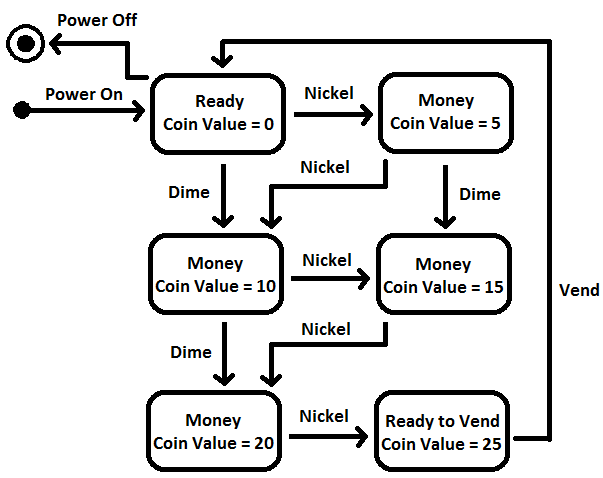
1. Explain the relationship between the cost of building software and the number of modules in the system.



1. Identify the following integration strategies.
   1. Take the entire integrated system and test it for the first time as a unit.
   * **Big Bang Integration** – throw everything in together and wait for it to explode
   1. Start by testing only the user interface with underlying functionality simulated by user.
   * **Top-down Integration**
   1. Testing the lower levels of the system via drivers.
   * **Bottom-up Integration**
   1. Can you do both top and bottom integration?
   * Yes, it is a better approach
2. What are verification and validation?

* **Verification** – Did we build the product right? Functionality.
* **Validation** – Did we build the right product? Meets requirements.

1. Create a state diagram to model the operation of a vending machine that sells a single flavor of soda for 25 cents. Only dimes/nickels can be used and the machines do not have coin return.



1. This is a screenshot from the running of an application. As the user enters a number, the program maintains an internal array to hold all of the values entered by the user. After each number is entered, the program adds it to the window of “data values” and updates the “average.” Assume that the designer has associated a class named Model with the storage of the values in the array. In addition, there is a Control class to get new numbers and two classes, View1 and View2, to take care of displaying the values and average, respectively. Draw a sequence diagram that includes one of each of the four objects above. In the diagram, show what happens.

* **“Life-lines” – the dots indicate the time; line has started and ended**

