**CS 4321 - Video Lecture Expectations**

**Video P4L1: Software Refactoring (22 minutes)**

Watch video P4L1 on Udacity. Answer the questions below and submit on Blazeview (HW-P4L1). Instructions:

* Do not remove the questions.
* You can provide the answer(s) where the blank is, but preserve the underline (or use a different color for the answers)
* Or, you can provide the answers below the questions. For example, you could type: Answer: x, y, z.

**Questions to be answered**

1. A \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ is made by a human and is the cause of a \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ which is an incorrect piece of code which can lead to a \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ which is an observable incorrect behavior of the software.
2. (The answer for this question comes from your own thoughts, it is not in the video) The official IEEE definitions of these terms is shown below. Many times, we use these interchangeably.

|  |  |
| --- | --- |
| **Term** | **Definitiion** |
| Failure | Observable incorrect behavior of the software. A system crash. A system bombs |
| Fault (Defect, Bug) | Incorrect piece of code. A flaw in any aspect of the system that contributes, or may potentially contribute to the occurrence of one or more ***failures***. A defect can be in code, in a requirement, design, or other places. |
| Error | An inappropriate decision, slip-up which causes a ***defect*** in a software product. |

Why do you think it could be important to differentiate these?

|  |
| --- |
|  |
|  |
|  |

1. List four approaches to verification.

|  |  |  |  |
| --- | --- | --- | --- |
|  |  |  |  |
|  |  |  |  |

Note:

1. The author didn’t say much about static verification. This is also called *static analysis* and *static program analysis (SPA)*. My understanding is that it is a program(s) that analyzes your source code without actually running the code. A simple example is Eclipse compiling your code continuously, or indicating that a variable or method is never used. However, there are many more sophisticated programs that can check all possible executions of the program and search for vulnerabilities. Static analysis has become popular. A list of tools for static code analysis:

<https://en.wikipedia.org/wiki/List_of_tools_for_static_code_analysis>

1. A very thorough treatment of SPA is found below. The development provides all the mathematical theory as well as applications. 100+ pages. <https://cs.au.dk/~amoeller/spa/spa.pdf>
2. The main advantage to testing is that it generates \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ while the main disadvantage is that it is \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_.
3. The main advantage to static analysis is that it considers \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ while the main disadvantage is that it can generate \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_.
4. Testing cannot prove the\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_, but only \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_.
5. \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ testing involves testing the individual modules in isolation

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ testing the interactions among various modules

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ testing involves testing the functional and non-functional requirements

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ testing involves validating that the system does what the customer wants it to

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ testing takes place every time we change the system

1. A \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ occurs when you make changes to code but the changes result in unrelated code being negatively affected.
2. \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ testing is one of the reasons software maintenance is so expensive.
3. \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ testing involves releasing the software to a set of selected internal users.
4. \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ testing involves releasing the software to external users.
5. \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ testing is where we consider the software as a closed box and is based on a description (specification) of the software and \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ testing looks at the actual code and tries to cover as much coded behavior as possible

Notes:

1. The author says a disadvantage to black-box testing is that it, “cannot reveal errors due to implementation details”. While that is true, I believe “errors” should be replaced by “fault”. We remember from a previous clip that he said that the developer is the only person who knows what the error is and that a “fault” is a buggy piece of code. As I see it, black-box testing reveals failures, but not the faults. In other words, when a failure is exhibited, we don’t know what the fault is (much less the error), we have to seek it out.
2. The author says that a disadvantage to white-box testing is that it, “can't reveal errors due to missing paths.” In other words, the specification may say the software can do X, Y, and Z, but perhaps only Y and Z are fully implemented. Thus, white-box testing wouldn’t be able to detect any errors in the functionality of X because it is not present.