HW#1 – Jinyang Li – CS495 – A20317851

Briefly discuss each of the following:

Why is it difficult, if not impossible, to prove that a developed system is defect-free before it is released?

Software testing is to identify as many as defects as possible, but not to identify all defects, also software testing is not a proof of correctness, it's all about the assurance that most of the defects have been identified thus reducing the probability of undiscovered defects that might occur in live environment.

Also, the budget is limited, timelines are defined, resource count is fixed etc. Even a simple text-box needs data tests, UI tests, security tests, compatibility test, etc.

Even if you got unlimited time and budget to do exhaustive testing, the environments are different, maybe a particular client operation system versions would cause bugs. Although your found 0 defects in development, there're still chances the defects occurs in product. So, before it is released, it's is very difficult to prove the system is defect-free.

Which do you think is more valuable with respect to your goal of producing high-quality systems – a competent programmer or a competent tester?

I think a competent programmer is more valuable since there is no "defect-free" software in real world, even a "hello word" program can come out defects in some environment. A good programmer means the system he programmed is DRY and follow the design principles such as SOLID etc. Then, the system is easy to maintain and flexible, also should be usable. Even there are bugs found in future, the system can easily fix the bugs to ensure high-quality. According to defects clustering, 80% defects occurs in 20% modules, a good programmer would not make the whole system to be rewritten if one of its module is fallen. Also, a good programmer, in theory can make 0 defects, and there is no need for testers. Thus, in real world, a good programmer can reduce the costs. I think a competent programmer is more valuable.

What is the significance of the "distance" between defect insertion and defect discovery?

An injected defect is adding defects to a system on purpose during testing, to test and verify that the system behaves as expected in those situations, which can insert any time, any amount, big or small.

The time, distance between inserted and it get discovered does matter since if the distance is too long, team may build something on defective fundamental system, thus the defect getting bigger.

The greater the distance, the big the trouble.

Like if we found requirement not good, we can found it because inserted defects were discovered. As time goes by, the under found defects would cost more money.

Consider the requirement "The data entry form shall be easy to use."

Rewrite the requirement such that it is more likely to be satisfied

Describe a test strategy for proving acceptability

The data entry form shall contain clearly, understandable label associated with well-organized input fields. So, user can input each field in 10 seconds, and a user who can read newspaper can filled out the form and understand label.

Test Case

Test if the form can notify user if input format or type is wrong and gives out example.

Test if each field of form can be completed in 10 seconds.

Test if every label is accurate and a widely-used word

Test if input fields are organized under a layer. Such as grid, horizontal.