**Summary of Ideal Testing and Non-Ideal Testing**

**Faye Van Roekel**

**Web-430 – Assignment 3.3**

Ideal Testing is a testing in development that is designed to catch bugs and errors in the early stages of automated testing. Ideal testing is broken up into a pyramid and then split into 3 stages.

1. Automated Unit Tests
2. Automated Service Tests
3. Automated User Interface Tests

Automated Unit Tests make up the base of the pyramid. Testing in this level allows us to “check our work”, getting immediate feedback and letting developers know exactly where the bugs are (Palamarchuk, 2017). The idea is that the test should only test a small amount of code which is independent code and isn’t dependent on anything else. If the test doesn’t work the tester will know where the problem is located (within the small amount of code) and the larger system will not be affected by the test. Most of the tests should be performed at this level since they are typically less expensive to maintain and are fairly easy to create.

The Automated Service Test is the middle layer of the pyramid. This level includes testing the API and Integration. Integration testing is a level of software testing where individual units are combined and tested as a group. The purpose of this level of testing is to expose faults in the interaction between integrated units. Typically, the service tests are more complex to create and are more expensive to maintain. Their value is in showing how two or more parts of the system will work together. There are different testing layers within the service tests. They can be divided into testing of components, integration and API.

Automated UI testing makes up the top layer of the pyramid and is the most comprehensive over the other tests in the pyramid. The system’s flow from end-to-end is covered throughout the Automated UI tests. Typically, they are the most expensive to develop and maintain. They are important since they show if the system works as a whole. The purpose of performing end-to-end testing is to identify system dependencies and to ensure that the data integrity is maintained between various system components and systems.

Non-Ideal testing in development is shown in an upside-down pyramid. Non-Ideal testing consists of Graphic User Interface (GUI) testing and manual testing on the top large layer of the pyramid. The center layer of the pyramid is integration testing. The bottom small triangle in the pyramid will have some unit testing. Non-Ideal testing usually doesn’t allow testing teams to have access to the code base needed to create the unit tests. This Non-Ideal testing is often used in businesses and organizations so often developers are not testing the code early in the process. The Non-Ideal testing approach seems to be top-heavy with services and UI testing. More time is spent on manual testing and less time on unit testing within this model.

If possible, the best approach seems to be using Ideal Testing. This is key to developing a high-quality application. The methods and principles used to accomplish this task are crucial. By using Ideal Testing, it allows much more testing on the base or the front end with smaller amounts of code and then build up into the integration and UI layers. It can show that all of the levels of the system will work together, and the product will be more secure and well built. By automating processes, the testing all runs automatically and consecutively which in turn will free up time for other tasks.

References:

Palamarchuk, S., (2017). Best Testing Practices for Agile Teams: The Automation Pyramid., How to Apply Test Automation Efficiently and Effectively with the Automation Pyramad., Retrieved 17 January 2020, from: <https://abstracta.us/blog/test-automation/best-testing-practices-for-agile-teams-the-automation-pyramid/>

Willett, J., (2016). The Evolution of the Testing Pyramid., Retrieved 17 January 2020, from: <https://james-willett.com/2016/09/the-evolution-of-the-testing-pyramid/>

Cochran, T. (2018, May 28). Test Pyramid: the key to good automated test strategy. Retrieved 17 January, 2020, from <https://medium.com/@timothy.cochran/test-pyramid-the-key-to-good-automated-test-strategy-9f3d7e3c02d5>