Jake Orben

Final

December 12, 2013

Dr. Pulimood

1.

The two moral imperatives “avoid harm to others” and “respect the privacy of others” are designed to protect the public from malicious software, intentionally or otherwise. This guideline is necessary because, as computer scientists we have access to a large amount of private data, such as social security and credit card numbers and it is our responsibility to use it properly. With such data, an individual could steal great deals of money as well as do harm to others. Furthermore, if many computer scientists took advantage of others, the field, as a whole would be looked down upon, leading to a decline in the industry as well as a lack of trust. Additionally, if a computer scientist does not do everything to protect a user’s information a problem could arise. For instance, if the Amazon database of passwords and credit card information were to be successfully attacked, thousands of users could simultaneously have their personal information stolen. The consequences of such an attack could be harsh; Amazon could lose customers due to such an attack, their stock prices could drop; Amazon could even be sued for liability if the issue is not handled properly and if customers are not informed promptly. An example of software that violates these moral imperatives is the PRISM program run by the United States government. The sole intent of this system is to collect data on the United States population, something entirely against the morals set forth by the ACM.

There are several ways in which personal information could be protected. Encryption is one of the most popular methods of data security. Encryption takes data and rearranges it according to a formula. The data can then only be decrypted if a user has the key for the file. Such technology is often employed on secured pages, such as a bank’s user interface page, where sensitive information is entered and viewed. Testing for software penetration is another way for a programmer to ensure that data is secure. This type of testing is often done by outside companies or individuals who are familiar with the week points in the security of the average software system. Finally, Software designers can implement systems that collect as little user data as possible, while still being functional. Such a design would decrease the amount of data stolen if there is a system breach.

2.

It is more cost and time efficient to plan a program ahead of time and perform tests on a system than coding a system immediately. Software testing provides information and statistics on the quality of a software solution. If a program is not designed properly, many issues could arise. For instance, if a group of programmers begins implementation with only a loose idea of what others are doing, the inputs and outputs of the sections being worked on might not match up. Furthermore, two programmers could, in advertently be working on the same section at the same time, causing a loss in productivity. Worse yet, each programmer could have differing ideas about how a system will be implemented, meaning that the functions created might not work in conjunction with one another or could produce unnecessary data. If this were the case, entire methods could be thrown away and wasted, causing a loss of both money and time. Any combination of these issues could cause you to receive a sub-par software solution, causing you to lose trust in your programmers.

Testing is a critical part of software development as well. Testing ensures that you get a robust and well-designed software solution. There are several issues that can develop as a result of not testing software. One issue is ‘crashing’, where, while processing data, the system stops working. This is often the result of bad user input, which can be tested for using methods such as Fuzz Testing, where a variety of strange characters as well as other inputs are used, in an attempt to stop a program from working. Another issue that can develop from lack of testing is a mismatch of input from one section of a program to another. Such an issue can develop when a program is badly designed and can be fixed using system testing, a test where all of the components of a system are tested as a complete unit. Such a test would ensure that all segments of the program work together as a whole. Finally, individual segments of a program might not work on their own, for instance, a method designed to multiple two numbers might divide instead. Unit testing is the best way to ensure that each function works. Unit testing is a type of testing that tests the individual methods and classes in order to ensure that data is processed properly. I would recommend that all three of the aforementioned tests be used. Fuzz testing to ensure that incoming data is handled properly, Unit testing to ensure that functions are performing properly and System testing to make certain that all components of a system are working cohesively.

As a whole, it is more efficient to plan out and test a software solution rather than to simply put code together in a patchwork manner. Redesigning a poorly implemented system that does not work would cost a great deal of time as well as resources. As a result, software solutions should always be planned out first and tested well once completed.