**COMP 5710 Presentation Abstract**

**Haden Stuart – has0027**

The reuse of software in this day and age can be an extremely efficient way to solve a problem, but at the same time it can be a major liability. As the internet continues to grow, the software that is implemented must also grow in order to fix new problems and prevent future bugs. This brings up the topic, “Is software reuse reliable when it comes to quality assurance?”. Most programmers in today’s world rely heavily on the internet and other developers for blocks of code to solve specific problems. This could include GitHub projects, imported repositories, or even Stack Overflow pages with solutions. While this is a great and efficient method at the time, you must be extremely careful since all software will decay with age.

Reusing software may seem like the best and most efficient route when attempting to find a solution but it also has many downsides. As programs are updated, bugs and vulnerabilities can be introduced that could be severely bad if implemented. If someone were to reuse some sort of code that had a vulnerability that they were unaware of, then the results could be terrible. Or Say you build a program around someone else’s imported library and they stop maintaining it one day. Now you are left with a broken program that you must go back and attempt to rework when you could have implemented something new and locally maintained from the start.

Aside from all the cons of reusing software, there are also many pros that must be stated. Reusing blocks of code or even old solution ideas can be extremely effective and efficient when attempting to solve a problem. When deciding to reuse a method or solution in a program you must always remember to evolve the reused code with the rest of the program constantly. This way, any code you reused will not cause any unnecessary bugs in future production. Something to think about in terms of “reuse” is knowing just how much code to reuse. Many code blocks that someone may want to reuse can be broken down into smaller pieces that can be implemented into new, maintainable code. This way the developer can reuse code efficiently while also keeping a reliable program that can evolve as time goes on.

**Sources:**

* The Challenge of Code Reuse; <https://www.perforce.com/blog/qac/challenge-code-reuse-and-how-reuse-code-effectively>
* Tips for Effective Software Reuse; <https://www.infoq.com/articles/vijay-narayanan-software-reuse/>
* ScienceDirect Software Reuse; <https://www.sciencedirect.com/topics/computer-science/software-reuse>
* Software Reuse Considerations; <https://accendoreliability.com/software-reuse-considerations/>