Week 6: Threaded Discussion I

**From your reading of chapter 7 and your own knowledge from other sources, discuss the following:**

If one person has written a component but others have revised it, who is responsible if the component fails?

What are the legal and ethical implementation of reusing someone else's component?

If one person develops a component, it goes through testing, and is iterated out into production, and then another developer comes along the following iteration and the component fails the next time it goes into production, then it seems as though the failure is the fault of the second developer; however, it is so easy, even with extensive testing, for a software bug to go out into production it wouldn't be fair to always pin the component failure blame on the last person who worked on it. It would probably be possible, through version management software, to look through the component's code history and determine who's fault it was; however, in my experience, if the software shop has the time to figure out blame, then it isn't a very productive software shop. Instead, organizations are better suited to adopt the more efficient practices of "egoless programming," where responsibility for component failure is something the whole organization shares, including the team of programmers who should have peer-reviewed each other's work, the quality assurance testers, and the managers who oversaw the whole process (Pfleeger & Atlee, 2006).

As for the legal repercussions of using someone else's component, if the component was obtained from a source external to the organization, the programmer has a responsibility to ensure the component has a valid open-source license, such as Apache, GNU General Public License, MIT, Mozilla, or some other standard (Nelson, 2006); otherwise, the programmer is stealing someone else's intellectual property. As for who holds the legal responsibility for a software failure, according to Security technologist Bruce Schneier, it does not appear that such a thing exists:

> It's the system that's broken. There's no other industry where shoddy products are sold to a public that expects regular problems, and where consumers are the ones who have to learn how to fix them. If an automobile manufacturer has a problem with a car and issues a recall notice, it's a rare occurrence and a big deal – and you can take you car in and get it fixed for free. Computers are the only mass-market consumer item that pushes this burden onto the consumer, requiring him to have a high level of technical sophistication just to survive (Schneier, 2008).

We cannot sue software companies for writing code with bugs in it, except in extreme cases, where it is found the software was purposefully written to be harmful, as with viruses and unfair business practices.

References:

Nelson (2006). *Open Source Licenses by Category*. Open Source Intiative. Retrieved from OpenSource May 24, 2009: http://www.opensource.org/licenses/category

Pfleeger, S., & Atlee, J. (2006). *Software Engineering: Theory an Practice*. New Jersey: Prentice Hall.

Schneier, Bruce (2008). *Software makers should take responsibility.* The Guardian, 17 July 2008. Retrieved from the Guardian May 19, 2009: http://www.guardian.co.uk/technology/2008/jul/17/internet.security

http://news.cnet.com/5208-1002_3-0.html?forumID=1&threadID=9323&messageID=67377&start=-1

http://www.ccsr.cse.dmu.ac.uk/conferences/ethicomp/ethicomp2002/abstracts/83.html

http://www.itproportal.com/portal/news/article/2009/5/11/eu-put-code-responsibility-software-makers-hands/

http://www.blackducksoftware.com/learn/role