Review 3

1/25/12

CS6V81.502

Seungtack Baek

**How Do Fixes Become Bugs?**

Whereas it is natural for bugs to appear from software, it is assumed the fixes for the bugs are bug-free. Yet, according to the authors in this journal, Yin, et al. the bug fixes themselves are not free from bugs. In fact, they can introduce new bugs (and they can be more severe than the previous bugs) or simply the bug itself can persist from incomplete fixes.

There are many reasons to why fixes are incomplete: time constraint, narrow scope, lack of knowledge of the buggy code. The researchers went beyond this. They also analyzed what types of bugs are more likely to introduce another bug then the other. They found that concurrency problems and memory management fault are the main types of bugs that introduce another bugs, thus “fixers” should be careful when they are fixing these types of errors. Also the authors suggest how to work with those types of errors.

I do see some room for improvements. What I wonder is why they would also use letter representation (like A, B, C or D) for commercial product. The authors mentioned that it is to protect their privacy and reputation, yet since they did not even told us what OS they are using, they could’ve make comparison between commercial OS and open source OSes.

**Question:**

1. They listed “lack of knowledge” of the code you’re working on as one of the biggest reason for incomplete bugfix. Also, they mentioned that it might be impractical to allocate one who wrote the code for the bugfix. In that case, what can we (or company) do?