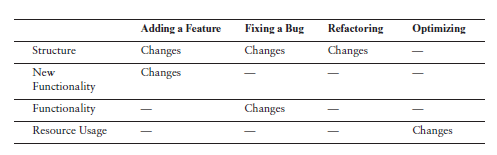
Working Effectively with Legacy Code-Notes

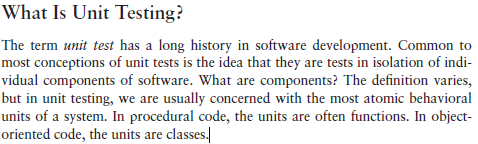
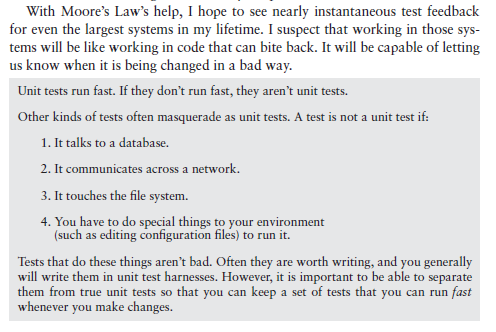
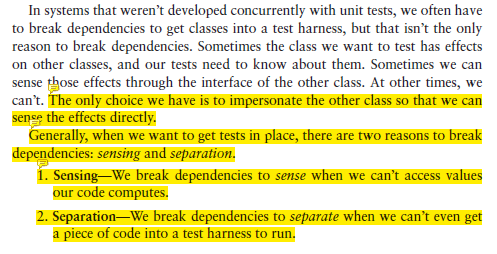
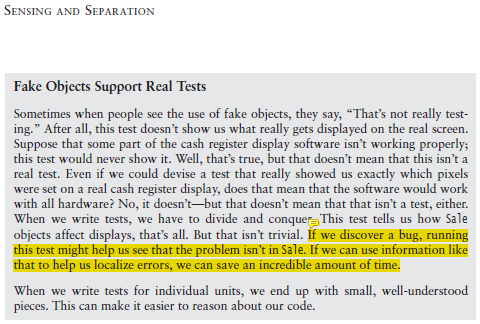
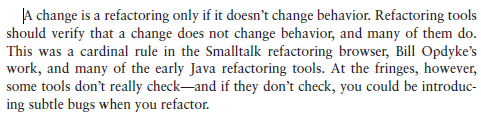
* **we can’t let “best” be the enemy of “better.”**
* Refactoring differs from general clean-up in that we aren’t just doing low-risk things such as reformatting source code, or invasive and risky things such as rewriting chunks of it. Instead, we are making a series of small structural modifications, supported by tests to make the code easier to change.

Optimization is like refactoring, but when we do it, we have a different goal.

With both refactoring and optimization, we say, “We’re going to keep functionality exactly the same when we make changes, but we are going to change something else.” In refactoring, the “something else” is program structure; we want to make it easier to maintain. In optimization, the “something else” is some resource used by the program, usually time or memory.

* 

Unit testing is one of the most important components in legacy code work. System-level regression tests are great, but small, localized tests are invaluable. They can give you feedback as you develop and allow you to refactor with much more safety.

* 
* 
* 
* 
* 
* 