Unit Tests

Lessons Learned

Unit tests are one of the corner stones of Extreme Programming (XP). But unit tests XP style is a little different. First you should create or download a unit test framework to be able to create automated unit tests suites. Second you should test all classes in the system. Trivial getter and setter methods are usually omitted. You will also create your tests first, before the code.

Unit tests are released into the code repository along with the code they test. Code without tests may not be released. If a unit test is discovered to be missing it must be created at that time.

The biggest resistance to dedicating this amount of time to unit tests is a fast approaching deadline. But during the life of a project an automated test can save you a hundred times the cost to create it by finding and guarding against bugs. The harder the test is to write the more you need it because the greater your savings will be. Automated unit tests offer a pay back far greater than the cost of creation.

Another common misconception is that unit tests can be written in the last three months of the project. Unfortunately, without the unit tests development drags on and eats up those last three months and then some. Even if the time is available good unit test suites take time to evolve. Discovering all the problems that can occur takes time. In order to have a complete unit test suite when you need it you must begin creating the tests today when you don't.

Unit tests enable collective ownership. When you create unit tests you guard your functionality from being accidentally harmed. Requiring all code to pass all unit tests before it can be released ensures all functionality always works. Individual code ownership is not required if all classes are guarded by unit tests.

Unit tests enable refactoring as well. After each small change the unit tests can verify that a change in structure did not introduce a change in functionality.

Building a single universal unit test suite for validation and regression testing enables frequent integration. It is possible to integrate any recent changes quickly then run your own latest version of the test suite. When a test fails your latest versions are incompatible with the team's latest versions. Fixing small problems every few hours takes less time than fixing huge problems just before the deadline. With automated unit tests it is possible to merge a set of changes with the latest released version and release in a short time.

Often adding new functionality will require changing the unit tests to reflect the functionality.

FullSize Collective Code Ownership

While it is possible to introduce a bug in both the code and test it rarely happens in actual practice. It does occasionally happen that the test is wrong, but the code is right. This is revealed when the problem is investigated and is fixed. Creating tests independent of code, hopefully before code, sets up checks and balances and greatly improves the chances of getting it right the first time.

Unit Tests provide a safety net of regression tests and validation tests so that you can refactor and integrate effectively. As they say at the circus; never work without a net! Creating the unit test before the code helps even further by solidifying the requirements, improving developer focus, and avoid creeping elegance.

Unit Tests XPprogramming.com JUnit.org

ExtremeProgramming.org home | XP Rules | Unit Test Framework | About the Author