Stop testing so much!

Press Release

Posted by:

Posted on : 2008/11/12 0:54:03

Automated testing is a great way of maintaining quality on a software project by providing quick feedback to developers when things break. Problem is, often teams find themselves with long-running suites of tests that become a time killer in the iterative development process. If the tests take too long to run, developers are less likely to run the full suite locally before a commit. Instead they commit their changes untested and rely on the Continuous Integration (CI) server to test it. In many teams this can mean the CI server gets quickly overloaded with changes to test, and developers wait hours to find out they broke the build.

With the release of Clover 2.4 we've added a new test optimization feature that can dramatically reduce build times by selectively running only the tests relevant to a particular change. This makes it practical for developers to run the test suite locally prior to a commit. It also means CI server throughput is greatly improved, both of which mean faster feedback to development teams.

When too much testing is ...probably too much

In many teams it can take far too long for the impact of a code change to be known by the submitting developer. The developer might wait many minutes or even hours before the Continuous Integration server gets to building and testing their change. If instead they've run the suite locally, their machine is tied up running tests, leaving the developer expensively idle.

Build breakages can often derail a whole development team, with all work grinding to a halt while the spotlight shines on the developer who introduced the problem as they attempt to fix it.

If a particular change is going to cause one or more tests to fail, the team needs to know about it as fast as possible, and preferably before it is committed.

Two approaches to smarter testing

So much of the testing effort is wasted because many tests are needlessly run; they do not test the code change that prompted the test run. So the first step to improving test times is to only run the tests applicable to the change. It turns out that in practice that this is a huge win, with test-run times dramatically reduced.

The second approach, used in conjunction with the first or independently, is to prioritise those tests that are run, so as to flush out any test failures as quickly as possible. There are several ways to prioritise tests, based on failure history of each test, running time, and coverage results.

Clover's new test optimization

As a code coverage tool, Clover measures per-test code coverage - that is, it measures which tests hit what code. Armed with this information, Clover can determine exactly which tests are applicable to a given source file. Clover uses this information combined with information about which source files have been modified to build a subset of tests applicable to a set of changed source files. This

set is then passed to the test runner, along with any tests that failed in the previous build, and any tests that were added since the last build.

The set of tests composed by Clover can also be ordered using a number of strategies:
* Failfast - Clover runs the tests in order of likeliness to fail, so any failure will happen as fast as possible.
* Random - Running tests in random order is a good way to flush out inter-test dependencies.
* Normal - no reordering is performed. Tests are run in the order they were given to the test runner.

Note that Clover will always run tests that are either new to the build or failed on the last run.

For more information please visit: http://www.atlassian.com/software/clover/?s_kwcid=HMClover