THE LEADING .NET CODE QUALITY PLATFORM

Building a Code Coverage Plan That Boosts Code Quality

September 2009

411 University Ridge, Suite 235
Greenville, SC 29601

Toll Free: 866.994.6676
Phone: 864.990.3717

www.ncover.com

NCOVER

Build a Code Coverage Plan that Eliminates Risk

Code quality is a topic that developers take personally. Poorly written code affects their livelihood, their reputation, and the time they spend on creating new solutions that people enjoy using and are willing to pay for. Taking a software product from idea to prototype to mass consumption is an amazing adrenaline-filled experience for a developer of any skill level. And yet, this process is not without issue. Bugs derail the entire process.

Buggy code is expensive to fix and saps development momentum. Developers hate to deal with it. No good developer plans to write bad code. But very few developers plan to not write bad code. A plan is absolutely necessary to achieve maximum code quality results—and it is not too hard to do using code coverage.

Code coverage analysis is uniquely able to demonstrate dramatic improvements in code quality because it touches multiple points of the development lifecycle.

While each function or role within the organization uses a similar interface, the resultant data can be interpreted most significantly in light of the objective currently being investigated Advanced features such as reporting and satisfaction thresholds also demonstrate overlapping flexibility for multiple disciplines to achieve complimentary results using the exact same data.

Most Developers Don't Want to Write Bad Code. They Just Don't Plan Any Other Option...

So Where Do You Start?

Integrating code coverage analysis is a great way to plan around not writing bad code. And it can be done in a series of non-evasive steps that not only accomplish the desired result of better quality code, but also drive efficiencies that limit overall software deployment risks and add significant profitability to the corporate bottom line. Understanding those potential risks is the first step toward creating a viable code coverage plan.

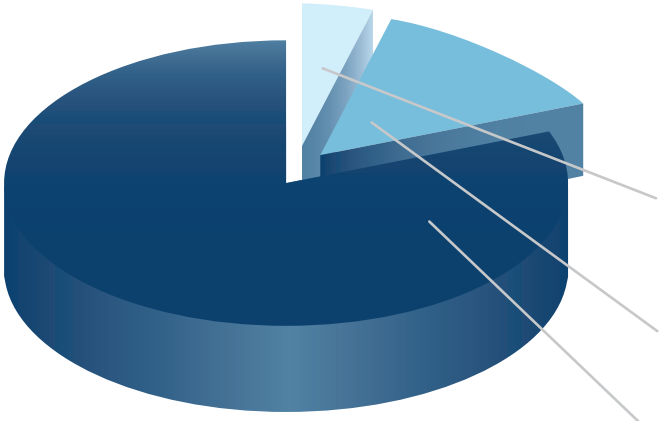
NCOVER

1 UNDERSTAND YOUR CURRENT DEVELOPMENT ENVIRONMENT

Each development environment is different. From the raw number of developers to concurrent projects being coded to the languages being utilized, each scenario is different. Taking the time to map out these obvious variables is absolutely necessary. Included should be the following, amidst other more functionally specific features:

- The current number of developers per project
- The complexity and iteration of releases per project
- The time spent from build to QA to deployment
- The desired personnel support for deployed project iterations and the cost to maintain that support.

Completely changing your environment is much more difficult to achieve than infusing core code coverage analytics into the environment. But to be effective at either option a clear understanding of the environment is absolutely crucial.



Based on a Developer's Typical 8-hour Day

2 INTEGRATE CODE COVERAGE AT THE DEVELOPER DESKTOP LEVEL

Code coverage products, like **NCover Complete**, are most helpful when they reside side-by-side with the code creation process. Research indicates that an exponential share of developer time is spent trying to understand the code in which he is currently working. According to one study, the average developer will spend up to fifteen times as much energy trying to figure out the details of his code than creating new project code.

This allocation of developer time is one reason why tools that provide advanced analytics around static code are so prevalent. Reducing the amount of time that it takes a developer to simply target suspected poor-performing code will go great lengths to boost personal efficiencies for the developer. The only way to realize those efficiencies is to have the code coverage platform directly on the developer's desktop where code coverage data can be gathered almost "real time" and improvements made before a build is deployed to the continuous integration environment.

Writing New Code
25.3 Minutes

Modifying Existing Code
75.8 Minutes

Understanding Code
378.9 Minutes

NCOVER

3 AUTOMATE ANALYSIS AND REPORTING IN THE BUILD ENVIRONMENT

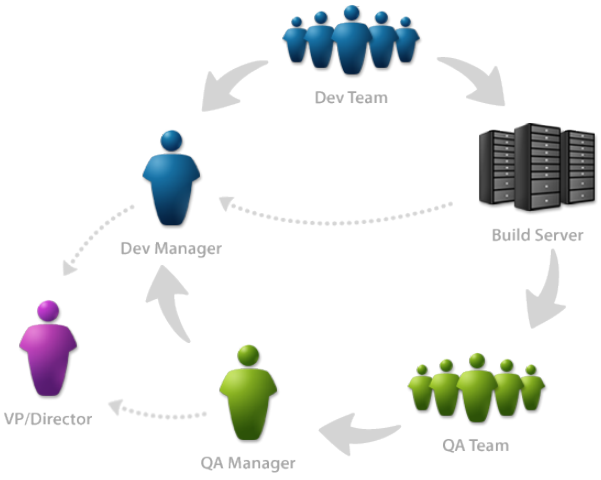
Once data has been interpreted at the desktop of each developer and any resultant steps finalized, the build will be sent to the continuous integration environment for incorporation into the existing and newly compiled code base. Code coverage analytics need to be wired into that framework to maintain code quality as a basic threshold for deployed software.

Comprehensive coverage analysis tools, such as NCover Complete, will provide advanced features that can facilitate even broader efficiencies for higher quality code creation. These features can seem intimidating at first, but provide a tremendous workflow through which iterative and consistently reliable software can be created.

This process can not be left to chance. You have to know what you want from your build before you get started. And, it's really not hard. Here are some of the factors that you will want to consider.

- The desired minimum percentage of code coverage necessary to "pass" a build
- The desired outcome from a "pass" or "fail" build, including the focused determinates for continuation or abortion of the build process
- The key reports that need to be automatically exported from the coverage process
- The network location for storing reports, logs, and process documentation for this part for the build process.

Once you have charted each of these features into a successful framework of intended outcomes, you will then need to implement and test each aspect for consistency and reliability. Since each build environment differs at least slightly from all others, there may be other reporting and performance nuances that you integrate into your workflow. The coverage scope of the tool being utilized will determine what additional features you utilize.*



* Products like NCover Complete can provide comprehensive analytics on static processes such as cyclomatic complexity. This along with run-time metrics like method visits and branch coverage can provide context for the healthy of a project code. Not seen in too many products but of unparalleled value is the ability to trend chronological code coverage results as a compelling gauge of future required support.

NCOVER

4 SUPPLEMENT QA PROCESS WITH INCEPTION FILTERING

The Quality Assurance division of any software development company is often structured around the tools used to analyze software rather than the process maintained to deploy healthy solutions. This variant in resultant expectations can lead to rifts between the coders building a structure of a project and the coders looking at the finished project for potentially buggy code. A properly implemented code coverage product can bridge this gap and consistently produce high-quality deployed solutions.

The most compelling QA efficiency derived from code coverage is to maintain a coverage threshold through which all builds sent to QA must satisfy. This will stop careless development efforts and redirect responsibility from the coder reviewing the project to the coder actually creating the project. You should be able to implement basic scripts to accomplish this.

By simply catching problems earlier, NCover Complete saves a company per developer $5,500 per year.

Another supplemental enhancement to QA can be found when complex debugging becomes necessary. Since leading debugging consultants deploy coverage analysis on a code base as a first step toward targeting potentially buggy areas of code, QA department will benefit from mimicking this behavior especially when confronted with otherwise unsolvable code problems.

5 REVIEW AND REFINE COVERAGE PLAN

The best plans still only seem to work "some of the time," so it makes sense that each party involved in the creation, integration, and review of the code coverage plan needs to play a role in evaluating the efficiency of that plan. Because code coverage analysis is so uniquely powerful in boosting value to an organization's bottom line, concerted effort needs to be applied in actually getting started. Even supplementing current processes with microimprovements can be the difference between happy clients or burned out developers.

After your plan is running, scheduling quarterly assessment meetings to refine data acquisition processes can lead to advancements in expected return-on-investment. Since risk mitigation and profitability are two natural evidences of a properly integrated code coverage plan, you will know definitively when you are getting the results you really want.

Take time now to lay the groundwork for your code coverage plan.

NCOVER

Why Should I Start with NCover Complete?

Harnessing a deep understanding of coverage analysis, **NCover Complete** empowers developers to predictably deploy quality software using the latest in multi-protocol coverage analytics and visual trending analysis. Studies indicate that end users of software created using the **NCover Complete** suite of tools appreciate the added performance and piece-of-mind from reliable solutions.

The end result of integrating **NCover Complete** into the development environment is seen quite clearly in higher quality code, quicker Quality Assurance problem highlighting, and faster product development timelines. Numerous research has pinpointed each of these areas as seeing significant benefit from code coverage instrumentation. Discussed less frequently but perhaps of even greater value is the bottom line benefit of code coverage.

Research has shown that the time and costs associated with fixing defects in software grow exponentially as the product moves beyond the development stage to the shipping life cycle. The generally accepted multiplier for this cost differential is a factor of 10 or 1000%. That means that a $1500 pre-shipping bug fix becomes a $15,000 disaster when discovered once the product is already in user's hands.

On average, developers implementing **NCover Complete** tools into their product development life-cycle will catch an additional ten bugs per developer per year in contrast to developers not using the product or their previous workflow before using the NCover approach.

The associated costs of fixing an average bug after the software had shipped to the customer were estimated to be $550. This compares to $55 to fix the same problem using **NCover Complete** in the development life-cycle. By simply catching problems earlier, **NCover Complete** saves a company per developer $5,500 per year.

By adding NCover to your product development life-cycle, you will be better positioned to improve your software reliability, to stabilize your quality assurance efforts, and increase the individual productivity of your developers. Most importantly, you will save money.

For more information about how NCover Complete can boost your bottom line right now, visit our website at www.NCover.com.