



The Software Development Challenge

The Trade-off between Time-to-Market vs. Quality

Table of Contents

1. Synopsis

2. Introduction

3. The Business Problem

4. The Impact of Poor Software Quality on Business

- a. The Business Impact
- b. The Developer Dilemma

5. Using Source Code Analysis to Make Testing Scale with Software Growth

6. Saving Software from Itself

- a. Coverity's Innovative Solution — Improving Quality
One Line at a Time with Automated Software Analysis
- b. Coverity Prevent
- c. Coverity Extend

7. What's Wrong with Current Software Tools and Processes?

- a. Runtime Testing Tools
- b. Static Tools
- c. Manual Code Reviews: eXtreme Programming (XP) or Scrum

8. Free Trial

9. About Coverity

1. Synopsis

Software developers are clearly facing a crisis. According to a recent *IDC Insight Report*, “Succeeding in software will not be about shipping great code but about going to market with the certifiably ‘perfect’ code that works out of the box as intended.” Customers are demanding higher-quality code — they are tired of the never-ending stream of patches and upgrades to software flaws that they believe should never have been there in the first place. According to a study commissioned by the National Institute of Standards and Technology (NIST), software errors are costing the U.S. economy an estimated \$59.5 billion annually. Even worse, more than one-third of these costs, or some \$22.2 billion, could be eliminated by an improved testing infrastructure that enables earlier and more effective identification and removal of software defects.

Software is everywhere and is becoming larger and more complex. In various industries, software has become a means to differentiate against competitors with extra features or to improve hardware efficiency by replacing mechanical components. For example:

Embedded

The average device now has a million lines of code and that number is doubling every two years.¹

Aerospace

A modern passenger jet, such as a Boeing 777, depends on 4 million lines of code. Older planes such as a Boeing 747 had 400,000 lines of code.²

Automotive

Recently, General Motor’s CIO, Tony Scott, stated that by the end of the decade, cars will be averaging about 100 million lines of code. That’s at least a tenfold increase since the early 1980s when the onslaught of computerized cars began.³

¹ *IDC* 2003

² *Technology Review*, 2002

³ *Internet News*, October 2004: <http://www.internetnews.com/infra/article.php/3428911>

The companies that succeed will be those that can develop high-quality software faster, more reliably and more cost-effectively than their competitors. That means delivering quality code that is low-risk and highly secure. Over five years ago, Coverity developed the first algorithms and research prototypes that could identify defects with a combination of precision, speed and accuracy that would meet the needs of even the most demanding industry customers as well as the open source community. Today, our platform has evolved into an enterprise analysis solution that scales to tens of millions of lines of code and thousands of software developers.

This white paper explores the critical issues that development groups face: the trade-off between quality vs. time-to-market, and risk reduction and management. It highlights the industry's only practical source-code analysis solution from Coverity. It covers:

- An overview of the current business challenge
- What's wrong with the current state of testing tools and processes
- The business impact of poor software quality
- Static source code analysis as an answer to the quality challenge
- Coverity's static source code analysis solutions
- Coverity's ROI

Coverity developed the first algorithms and research prototypes that could identify defects with a combination of precision, speed and accuracy.

2. Introduction

Software defects are inevitable. According to an article in *The Economist* (2003), "People who write software are human first and programmers only second — in short, they make mistakes, lots of them." As a result, most customers are frustrated with software and view failure as the norm, not the exception.

Granted, there is a high cost to producing highly reliable and secure software. However, the alternative may be even more costly. Lost sales and patch costs can directly bear high penalties, but what kind of price tag can be put on damaging the company's reputation or compromising competitive advantage? Consider these statistics:

- For every thousand lines of code output by commercial software developers, there could be as many as 20 to 30 bugs on average.⁴
- Worse yet, in 2003, *Developer News* reported that 50 percent of all bug fixes are done incorrectly the first time, often introducing new bugs in the process.
- As bugs progress through the development cycle, defects found become exponentially more expensive to fix — it is at least 30 times more costly to fix software in the field versus during development.⁵

With software bugs now costing customers and vendors tons of billions of dollars every year and exploited software vulnerabilities making headlines (e.g., the worldwide Internet slowdown in January 2003 from Slammer), companies are taking a closer look at their software development processes. According to the Cutter Consortium, more than a third of the companies surveyed in a recent study admit to releasing software with “too many defects.”

Software is found everywhere: embedded in missile defense systems, automobile navigation systems, airplanes, scanners, computers, cell phones and even toys.

3. The Business Problem

Why is software quality such a big deal? Software is found everywhere: embedded in missile defense systems, automobile navigation systems, airplanes, scanners, computers, cell phones and even toys. It is also found in applications fueling complex business processes, next-generation interactive TVs, communication infrastructures and across the internet underlying services.

According to *The Economist* (2003), “In a society dependent on software, the consequences of programming errors are becoming increasingly significant.”

⁴ *Baseline* magazine, March 2004

⁵ Software Engineering Institute at Carnegie Mellon University, 2001

What are the challenges?

Growing business complexity

There is no doubt that the Internet, Web services, new software categories, multiple channels of distribution and increasing device access (laptops, PDAs, cell phones) have changed the rules of the game. Today, businesses depend on software to make their processes more efficient.

Heightened competitive and market pressures on software development

Software projects are subject to rampant changes, such as specification updates and design modifications required to meet ever-changing market demands. Ultimately, the race to bring more functionality to market in less time too easily compromises quality.

Increasing market demand for software applications

A range of sectors, from consumer electronics and appliances to space, defense, automotive, avionics, telecommunications and medical devices. In addition, the size and complexity of embedded systems have increased, often requiring complete, multi purpose software applications. According to industry sources, the amount of embedded software doubles every 18 months, which puts a premium on reliability and robustness.

Inability of software development tools to keep pace with the growth of software complexity

While the market for development tools is growing at a rapid pace, most tools are still considered fairly primitive. Most tools, if they actually work, provide benefits in the late stages of the software development cycle when changes or fixes bear high costs. Furthermore, many of these tools involve human monitoring and intervention, and quickly run into scaling problems. This issue is aggravated by a lack of quality metrics, forcing managers to make critical resource-allocation decisions based on sparse and inaccurate data.

Increasing security concerns

The advent of the Internet has given rise to a profound increase in operational risk. Security problems such as cross-site scripting and SQL injections result from poor quality code, creating even more opportunities for attackers to threaten privacy and steal data. Security is fast becoming a top-priority concern for companies worldwide. According to *IDC*, enterprises will have spent more than \$16 billion for software security products by 2008.

Developers must create high-quality code with complex technical and business requirements and get to market quickly to maintain competitive advantages. While conventional efforts to deal with this complexity are being implemented across the industry, most industry analysts believe that the benefits of these process changes will take years to fully realize, if indeed they are successful.

4. The Impact of Poor Software Quality on Business

Defects that go undetected until after a product has shipped can cost the software maker many tens of thousands of dollars each to address and patch. The cost suffered by the end users is often orders of magnitude higher.

In business terms, poor software quality negatively impacts customer satisfaction, revenue and costs. For developers, poor software quality means more time fixing bugs and defects, not creating better, more innovative software.

A Solid Track Record

Our products have been tested and proven on open source solutions such as Linux, Apache, Berkeley DB, Mozilla and many others. With one of the largest install bases of any source-code-analysis solution, Coverity has helped improve code quality in a range of industries, from networking to gaming, security, embedded systems, online service providers, wireless and enterprise software vendors. Customers include Juniper Networks, Synopsys, NASA, Symantec, Palm and Wind River among many others.

4a. The Business Impact

The adverse affect of software defects has been well chronicled in textbooks as well as in recent studies such as the 2002 NIST report on the economic impact of software errors. Whether developing software for internal or external deployment, poor quality software hurts business in four main ways:

Decreases customer satisfaction

Common across all software disciplines is poor customer satisfaction.

Reduces revenue

Inability to up sell or cross sell, lost sales and loss of repeat business.

Increases operational and administrative support costs

Includes support costs such as warranty, replacement, litigation, patch development and lost worker productivity.

Delays time-to-market

This results in missed market windows and lost competitive advantage.

4b. The Developer Dilemma

The 2002 NIST study showed that nearly 80 percent of a developer's time is spent fixing bugs.⁶ (The typical programmer writes between 8 and 20 lines of code a day; the rest of the day is spent on debugging.) As a result, productivity has plummeted. Approximately 40 to 50 percent of a programmer's efforts are spent on avoidable rework. "Avoidable rework" means fixing difficulties with the software that could have been avoided or discovered earlier and less expensively.⁷ Peer reviews catch only 60 percent of the defects at best.⁸ Current popular methodologies such as eXtreme Programming are helpful, but they fall short of making an overarching impact on the end product.

According to Wind River, developers also struggle with a complex mix of architectures, middleware, development tools and operating systems. As a result, with embedded software 54 percent of software designs are late, 66 percent come in over budget and 33 percent fail to meet performance expectations. Understandably, faced with these metrics, programmers aren't a very happy lot, suffering from burnout, high stress and a negative work environment.

Coverity's engineers have devised solutions to tackle many of the difficult problems that have traditionally hampered source code analysis — build integration, compiler compatibility, high rate of false positives and effective root-cause analysis.

5. Using Source Code Analysis to Make Testing Scale with Software Growth

While at **Stanford University**, Professor Dawson Engler and four Ph.D. students in the Computer Science Laboratory set out to dramatically improve software quality as part of a massive research initiative. Their breakthrough work on meta-compilation opened new doors for source code analysis. Their new approach to source code analysis proved that high-quality results could be obtained over large code bases in a fraction of

⁶ <http://www.nist.gov/director/prog-ofc/report02-3.pdf>

⁷ <http://www.cebase.org:444/www/defectreduction/top10/top10defects-computer-2001.pdf>

⁸ Ibid

the time required by existing, best-of-breed research tools. Today, Coverity continues to innovate with new methods for improving software quality and security, developing innovative analysis technology that is generations ahead of the competition.

Coverity has developed the first system that can identify defects with a combination of precision, speed and accuracy that is enabling software developers to tame the size and complexity of today's code. In addition, Coverity's engineers have devised solutions to tackle many of the difficult problems that have traditionally hampered source code analysis — build integration, compiler compatibility, high rate of false positives and effective root-cause analysis. In summary, Coverity's products have been hailed by its customers as the most effective source-code-analysis solutions available in the market today because of their:

Scalability

Coverity scales to millions of lines of code while taking hours to run, not days or weeks.

Reliability

Coverity uncovers critical defects with a minimal number of false positives and false negatives.

Flexibility

Coverity can be easily retargeted to different build environments, code bases and bug types.

Simplicity

Coverity requires no changes to the code, build scripts and reports defects in a clear, easy-to-understand and actionable manner.

Is Static Analysis Right for You?

A Quick Guide to the ROI and Total Cost of Ownership (TCO) of Static Analysis

Software developers are the most rational buying group in the history of capitalism. Ironically, the ROI of static analysis is rarely articulated beyond the platitude “find more bugs sooner.” Software products—and quality requirements—differ dramatically, ranging from mission-critical heart monitors to tetris.

The ROI

The fundamental ROI of static tools centers on detecting—the defects developers and testing tools miss that can get into the field. To be effective, static analysis should provide an accurate analysis to detect deeper, more severe and hard-to-find bugs that are the relevant defects that developers need to eliminate.

The impact of finding defects earlier in the development process breaks down to reductions in several key areas:

1. Engineering costs It is estimated that 80 percent of a software engineer’s time is spent fixing defects for code in development or trying to repair bugs in the field (NIST 2002). Static tools should help developers minimize the time spent on defects by automating discovery.
2. Quality assurance costs Static tools should provide consistent, reproducible results during development that help deliver a reliable software product to QA teams. In turn, QA becomes more efficient because they inform their find on other types of testing usability such as testing or bad testing.

continue next page

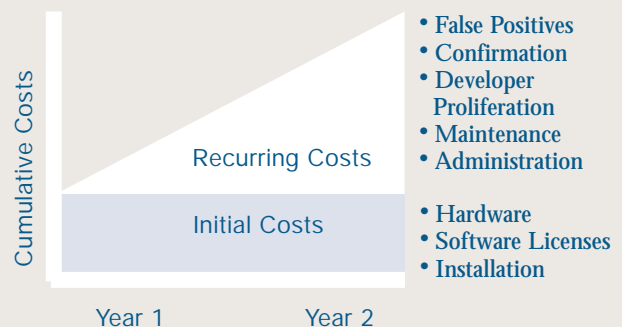
The other side of the equation: TCO

The total cost of ownership (TCO) is often the most overlooked and least understood aspect of static analysis deployment.

Initial costs

Typically, the initial costs of owning static tools are small:

1. Hardware Any additional processing hardware required to run the static analysis software
2. Software licenses The initial software license cost
3. Installation The length of time required to install the software

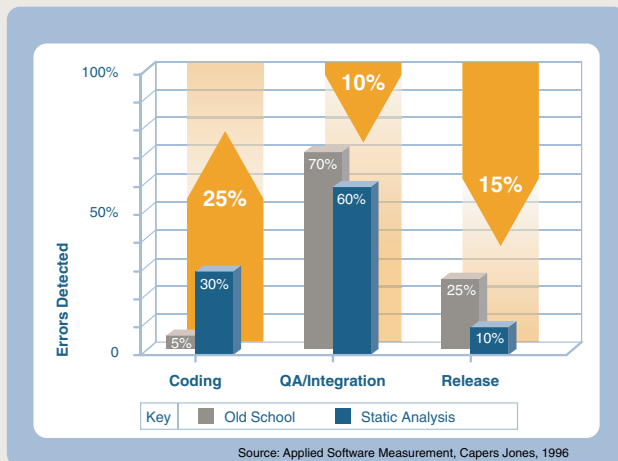


continue next page

The ROI

continued

3. Patch costs Static tools should help to eliminate the time engineers spend in the field due to poor software. One of the biggest costs of software defects is the unplanned, unallocated time writing and testing patches to software that did not work in the field.
4. Support costs In some organizations, a certain percentage of support time is spent dealing with faulty software. Static tools that discover complex defects and corner cases should see a fall in software-related support calls.
5. Warranty costs Poor software can lead to customers invoking expensive warranties. For developers writing software for devices automobiles or devices, this is often the largest potential cost.
6. Denial of service costs For service providers relying on software systems to deliver services, outages often have a lost-revenue cost associated with going down.



The other side of the equation: TCO

continued

Recurring costs

The true costs of static tools are the long-term costs associated with usage and deployment.

1. False positives False positives comprise the highest cost of static tools. If developers become overwhelmed by too many false positives (or “alerts”), static analysis rapidly devolves into shelf ware.
2. Configuration To minimize costs, static tools should work quickly. Ideally, deploying static analysis should be a simple, out-of-the-box process. Even within large development organizations, coding styles and standards change. Is reconfiguration required for every new software project? Are professional services required for every configuration?
4. Likelihood of adoption A static tool must be easy to use; otherwise developers will not adopt it. A key element to ensure widespread use is an intuitive and useful GUI that provides detailed root-cause analysis of a defect.
5. Maintenance Maintenance fees associated with the software license, often 15-20 percent of license cost.
6. Administration This is the number of full-time employees required to manage the static analysis tool.

Software developers are the most rational buying group in the history of capitalism.

6. Saving Software from Itself

Coverity's technology has already proven to be a powerful way to manage the growth of large and complex code. Many of the leading software companies in the world are currently realizing the benefits of delivering more stable, secure products on an accelerated development schedule using Coverity's solutions. The key benefits include:

Improving Time-to-Market

Coverity helps companies get differentiated products to market faster while simultaneously improving product reliability.

Reducing Risk

Coverity minimizes product malfunctions, recalls and warranty claims and improves security, thereby increasing developer productivity and your company's ROI.

Increasing Customer Satisfaction

Using Coverity results in better code, which translates into better products, a decrease in support costs and happier, more loyal customers.

Coverity and Linux

Coverity's experience with the Linux source code began when the founding team was working at **Stanford** on a research project then known as the Meta-Compilation Project. Focused on proving that compiler technologies could be successfully applied to find defects in robust software systems, Coverity's founding team developed an innovative engine that could detect bugs based on assumptions inferred from the source code. Early prototypes of Coverity's technology were applied on various versions of the Linux kernel, detecting thousands of defects and security holes.

6a. Coverity's Innovative Solution — Improving Quality One Line at a Time with Automated Software Analysis

“An army of engineers couldn't find this many defects.”

— Oleg Kiselev, Director of R&D at VERITAS Software and Coverity customer

Coverity can find bugs quickly and accurately for a code base consisting of several millions of lines of code with 100 percent path coverage—in only hours and all at compile time. Additionally, Coverity:

[Does not require test cases or code execution](#)

Provides consistent, reproducible compile time results that help deliver a reliable software product to Quality Assurance (QA).

[Delivers high-quality results](#)

Low rate of false positives, typically four real bugs to one false positive.

[Requires no changes to the development process](#)

Coverity integrates with any development environment. Coverity's technology does not require changes to your code or to your Guild process.

[Pinpoints defects with precision](#)

Coverity reports defects introduced in the latest build via email or web GUI, including defect quantity, location, distribution and severity.

[Addresses a class of errors that testing cannot, particularly security holes](#) For example, Coverity can inspect corner cases and unintended inputs.

6b. prevent™

Coverity's Prevent represents a breakthrough in source code analysis. By applying a combination of innovative, groundbreaking analysis techniques with innovative system architecture, we were able to produce the most effective analysis system ever built. Prevent is designed as a flexible analysis platform with modular plug-ins that implement specific simulations of the source code and report specific types of defects. Because of this modular design, each analysis is finely tuned to the type of bug that it detects, allowing Coverity's system to achieve a false positive rate that is unique among source code.

6c. extend™

Coverity's Extend enables developers to define and enforce unique coding rules. Coverity Extend leverages Coverity's analysis engine to automate thorough code audits—making these audits a simple, routine practice.

Sample Problems Coverity Detects

Crash-Causing Defects	<ul style="list-style-type: none">Null pointer dereferenceUse after freeDouble free
Incorrect Program Behavior	<ul style="list-style-type: none">Dead code caused by logical errorsUninitialized variablesErroneous switch cases
Performance Degradation	<ul style="list-style-type: none">Memory leaksFile handle leaksCustom memory and network resource leaksDatabase connection leaksMismatched array new/deleteMissing destructor
Improper Use of APIs	<ul style="list-style-type: none">STL usage errorsAPI error handlingAPI ordering checks
Security Vulnerabilities	<ul style="list-style-type: none">Array and buffer overrunUnsafe uses of tainted data
These defects are found with	<ul style="list-style-type: none">High accuracy; false positive rates are often below 20 percent100 percent path coverageCross-module, cross-function analysis to identify complex errors involving multiple components

7. What's Wrong with Current Software Tools and Processes?

Primitive. Insufficient. Inflexible. Unable to scale to even hundreds of thousands of line of code.

With errors often detected late in the software development process, the costs of testing have skyrocketed. Industry analysts report that the cost of testing alone comprises up to 50 percent of total development costs.

Here are some of the options conventionally used to find errors and the problems with each approach.

Runtime tools are only as good as the test cases that exercise the code.

7a. Runtime Testing Tools

Runtime tools such as Purify are vital testing tools. However, they are only as good as the test cases that exercise the code. Put another way, they will only find bugs in parts of the code that are executed. With the growth in the size and complexity of software, the benefits of runtime tools have eroded along with test coverage. Furthermore, certain kinds of software, such as embedded code, make it difficult to test many of the intended behaviors of the code at runtime since they rely on external elements or devices that aren't available or reproducible in the test lab. At best, even if defects are discovered with runtime tools, it is usually late in the development phase, when errors are already expensive to address.

7b. Static Tools

The main problem with conventional static tools is that they either do too little to ensure scalability or they try to do too much and end up not scaling. The tools that work on commercial level code bases detect superficial artifacts such as dangerous coding practices or syntactic programming rules. Rather than understanding the causes behind crash-causing defects, these tools "search" the code at a superficial level to detect hints, not root causes. On other side of the spectrum, tools that try to understand the code at a deeper level may run on code bases in the thousands of lines of code, but do not scale to code bases in the millions of lines.

7c. Manual Code Reviews

Studies illustrate the efficacy of manual code reviews—often reducing software errors dramatically. However, these techniques emphasize people and interactions over tools and processes, and discourage the use of tools for tools' sake. They rely on intensive reviews that are extremely labor-intensive and difficult to perform accurately. With eXtreme Programming, a programmer literally watches over someone's shoulder while they are coding, trying to spot the bugs. In addition to the expense of the extra resources, it is still a human process prone to errors. For example, peer reviews can expect to find only around 60 percent of defects within code.⁹

⁹ *Baseline* magazine, March 2004

8. Free Trial

“With Coverity, we discovered bugs in mature, stable code believed bug-free for years. These bugs hadn’t been detected by our test processes or encountered in the field.”

— SYMANTEC

To find out more or to get a free on-site trial, call us today.

9. About Coverity

Coverity makes the world’s most advanced and scalable static-source-code-analysis solution for precisely identifying software flaws and security vulnerabilities. Coverity was founded in 2002 by leading Stanford University scientists whose four-year research project led to a breakthrough in solving one of the most difficult problems in computer science. That research breakthrough allows developers—while they program—to quickly and precisely eliminate software flaws and security vulnerabilities in tens of millions of lines of code. Today, Coverity’s solution is used by many leading companies to significantly improve the quality of their software, including Synopsys, Juniper Networks, McAfee, NASA, France Telecom, Palm, Sun Microsystems and Wind River.

Contact Coverity

<http://www.coverity.com>

sales@coverity.com

185 Berry Street Suite 2400

San Francisco, CA 94107 USA

Phone: (800) 873-8193



185 Berry Street Suite 2400
San Francisco, CA 94107 USA
Phone: (800) 873-8193

<http://www.coverity.com/>
sales@coverity.com