

Controlling Software Complexity

The Business Case for
Static Source Code Analysis

Ben Chelf, Coverity CTO
Andy Chou, Coverity Chief Scientist

Introduction

Software developers today face significant opportunities and challenges. The appetite that both organizations and consumers share for software has made development a dynamic and competitive business, creating opportunities for large companies and start-ups alike.

Because of the increasing role that software plays in almost every facet of our lives and businesses, developers are under constant pressure to perform. Today, developers are challenged to deliver increasingly advanced applications on stringent timetables. Compound this with the growing complexity of applications themselves, add distributed or outsourced development teams, and it's easy to see why software is becoming larger and more complex to produce and manage.

According to an 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.

A study commissioned by the National Institute of Standards and Technology (NIST) reinforces this idea—the study found software errors cost the U.S. economy an estimated \$59.5 billion annually. The same study notes that 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.

In many industries, software is creating a new layer of competition as companies strive to deliver more features and functionality, with higher quality and security control—while at the same time asking developers to meet more aggressive delivery dates. For example:

Embedded

The average device now has one 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 only 400,000 lines of code.²

Automotive

General Motors' former CIO stated that by the end of the decade, cars will average about 100 million lines of code—a tenfold increase since the early 1980s when the introduction 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>

Defects and Software Quality

An experienced programmer knows that software defects are inevitable. They are a natural part of the evolutionary process that most applications experience. Whether you're working on the initial version of an application or expanding an existing application—defects happen. Today, new technology makes it possible to eliminate much of the pain and frustration that can come from hunting for defects in your code.

While in principal, producing highly reliable and secure software is the goal of every developer, the cost of code defects makes security and reliability mandatory. Flawed software translates to lost sales and patch costs, which can create significant financial penalties for companies. Taking the price of software failure one step further, we can only theorize about the impact on a company's finances and competitive advantage due to damaged reputation caused by software failures. 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.
- As they 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 according to a 2007 Forrester report.⁴

Because software bugs cost customers and vendors billions of dollars every year and exploited software vulnerabilities are consistently making headlines, companies need to take 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.”

⁴ *Managing Application Security from Beginning to End*, Chenxi Wang, Ph.D., 2007, Forrester

The Business Challenge of Software

Why is software quality becoming a larger issue over time? Because software is everywhere: 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's underlying services. With this growing reliance on technology, development organizations are now faced with new business challenges related to code quality, such as:

Growing business complexity

Businesses depend on software to make their internal processes more efficient, from the Internet and Web services to new enterprise software categories that include multiple channels of distribution. At the same time they rely on increasing device capabilities (laptops, PDAs, cell phones) to enable almost all fundamental business processes.

Heightened competitive and market pressures

Software projects are subject to rampant changes, such as specification updates and design modifications required to meet evolving market demands. Ultimately, the race to bring more functionality to market in less time can compromise quality.

Increasing market demand for multi-threaded software applications

A range of market sectors, including consumer electronics, defense, automotive, avionics, telecommunications, and medical devices have new, heightened appetites for embedded multi-threaded software to take advantage of multi-core hardware. According to industry sources, the amount of embedded software doubles every 18 months, placing a premium on software performance.

Development tools lag behind software complexity

While the market for development tools is growing at a rapid pace, most tools are considered to be fairly primitive by developers. Generally, these tools tend to provide value in later stages of software development, but by that point defects have already inflicted a high cost. Further lowering their ROI, many of these early tools suffer from high false positive rates and can quickly run into scaling problems for large code bases.

Increasing visibility of security failures

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, and create even more opportunities for attackers to threaten privacy and steal data. According to IDC, enterprises will have spent more than \$16 billion for software security products by 2008.

The Impact of Uncontrolled Software Complexity

Defects that go undetected until after a product has shipped can cost development organizations hundreds 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. While for developers, poor software quality means more time fixing bugs, not creating new applications. The adverse effect of software defects has been well chronicled in textbooks and 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:

- Decreased customer satisfaction—Common across all software disciplines.
- Reduced revenue—Inability to up-sell or cross-sell, lost sales, and loss of repeat business.
- Increased operational and administrative support—Warranty, replacement, and litigation expenses, as well as patch development and lost worker productivity.
- Delayed time to market—Resulting in missed market windows and lost competitive advantage.

The 2002 NIST study reported that developers spend nearly 80% of their time fixing bugs. By extension, this means 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, developer productivity has plummeted. Approximately 40 to 50% of a programmer's efforts are spent on avoidable rework, i.e. fixing difficulties with the software that could have been avoided or discovered earlier and less expensively.

How to Control Software Complexity

Coverity offers state-of-the-art static source code analysis technology that finds critical defects and security vulnerabilities in C/C++ and Java source code. Coverity's products automate the detection of these defects in complex software by compiling and analyzing the code at build time.

Today, Coverity Prevent SQS commands the largest installed base of any source code analysis solution in the market. Coverity has played a vital role in improving software quality and security in industries such as networking, security, embedded software, medical devices, telecommunications, wireless and enterprise software.

Coverity enables development teams to quickly identify critical defects that can compromise software performance due to crashes, security vulnerabilities, memory corruption, performance degradation, and unexpected behavior. By helping companies improve code quality and security, Coverity's static analysis solution decreases time to market and optimizes developer productivity.

To date, Coverity has analyzed more than one billion lines of mission-critical product code, ranging from scalable database systems to safety-critical embedded systems used in the most fault-intolerant deployment environments.

In addition to commercial expertise, Coverity also has an exclusive relationship with the U.S. Department of Homeland Security to help defend widely-used open source software against potential security attacks. For more information on this initiative, please visit <http://scan.coverity.com>.

Coverity's products were initially tested and proven on open source code bases such as Linux, FreeBSD, MySQL, Apache, and Mozilla. That analysis has resulted in numerous high-priority defect and security advisories that were quickly addressed by the open source community.

Coverity continues to drive innovation in the field of static analysis. The company introduced the world's first use of Boolean satisfiability in software analysis. This innovation enabled the use of sophisticated SAT-solvers to analyze software code. Also this year, Coverity introduced the most advanced use of concurrency checkers to test for dangerous, hard-to-find, race-condition defects.

Behind these advances are the people of Coverity. More than 30% of Coverity engineers have Ph.D.s in computer science, enabling the company to deliver the world's most sophisticated solution for automatically identifying and eliminating software defects.

Return on Investment (ROI) and Total Cost of Ownership (TCO)

Software developers are possibly 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.

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.

Finding defects earlier in the development process reduces costs in several key areas:

Engineering costs

It is estimated that 80% 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.

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.

continued next page

TCO

The total cost of ownership 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:

Hardware

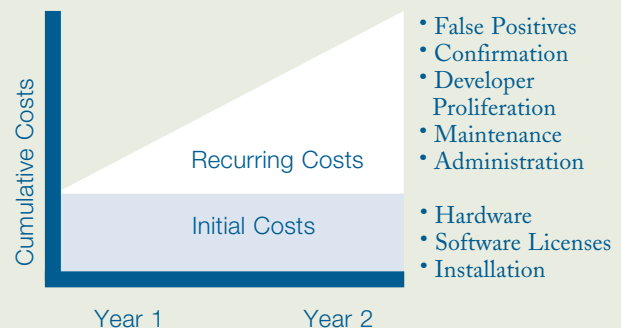
Any additional processing hardware required to run the static analysis software.

Software licenses

The initial software license cost.

Installation

The length of time required to install the software.



continued next page

ROI

continued

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.

Support costs

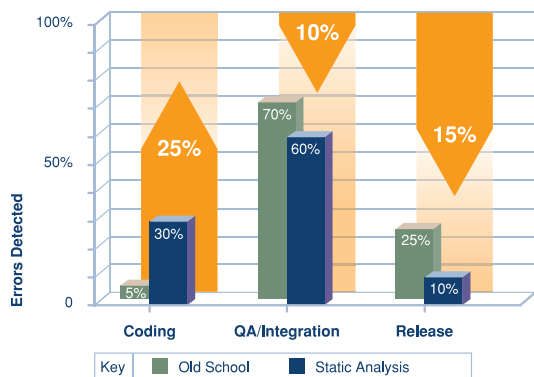
In some organizations, a certain percentage of support time is spent dealing with faulty software. Static tools that discover complex defects and corner cases should see a fall in software-related support calls.

Warranty costs

Poor software can lead to customers invoking expensive warranties. For developers writing software for automobiles or devices, this is often the largest potential cost.

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.



Source: Applied Software Measurement, Capers Jones, 1996

TCO

continued

Recurring costs

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

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.

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?

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.

Maintenance

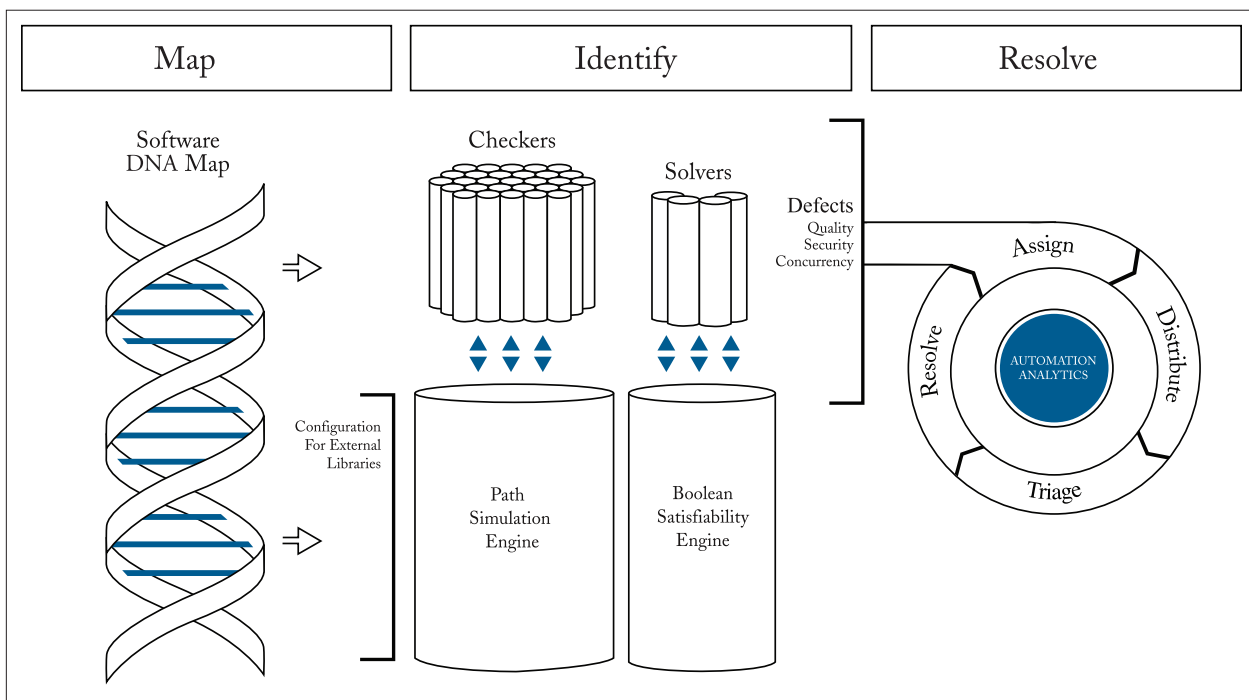
Maintenance fees associated with the software license, often 15–20% of license cost.

Administration

This is the number of full-time employees required to manage the static analysis tool.

Coverity Prevent™

Coverity Prevent checks 100% of the paths and values in C/C++, and Java software projects. Our unique combination of analysis engines based on path simulation and Boolean satisfiability provides the most sophisticated analysis of your software, including dependencies, key third-party libraries, and projects spread across multiple development groups. Coverity's low false positive rates; ability to find critical, must-fix errors; and defect resolution tools make developers' lives easier and improve their ability to find and fix defects. Here is the basic product architecture:



Software DNA Map

Coverity Prevent is the only static analysis solution to watch every operation of the build, not just calls to the compiler. This patent-pending technology insures the integrity of your development environment because we make zero changes to your build regardless of the type of build system. Be cautious of vendors that offer different solutions for different build systems, as these can lead to inconsistent feature sets and difficult implementations.

Coverity Prevent tests a representation of your software in the same manner as compilers testing their output for correctness. For every source file, we compile it just like a compiler—in fact, our compiler is so precise, we verify our representation of the software by leveraging existing compiler test suites to make sure our representation of your source code is semantically equivalent to the source code for every file in your code base.

Because Coverity creates the most accurate representation of your software available, we can deliver the most accurate defect detection results.

Software Analysis

Coverity Prevent automatically identifies critical defects by analyzing a Software DNA Map of your code with the most sophisticated analysis engines available today. By leveraging a combination of Path Simulation and Boolean satisfiability (or SAT) engines, our groundbreaking technology provides 100% path coverage and 100% value coverage for your code base. Importantly, Coverity Prevent can provide this while scaling to analyze millions of lines of code in just a few times your normal build time.

This provides you with the most accurate and precise analysis of your code available today, with the lowest false positive rates in the industry. Coverity customers average false positive results around 15%, with many of our customers experiencing out-of-the-box false positive results in the 5–10% range.

Because Coverity automatically identifies software defects with the lowest false positive rate in the industry, we deliver a solution from which developers can immediately benefit.

Defect Resolution

Coverity Prevent also provides a workflow process and GUI that seamlessly integrates into your existing environment to help resolve defects in a predictable and timely manner. Based on our work with more than 350 customers, we understand development processes and how static analysis can and should be used.

Coverity Prevent is the only solution that automatically assigns ownership for defects to the appropriate developer within a comprehensive workflow that mirrors your existing process for dealing with defects. The Coverity Professional Services team is also available to customize your deployment in a few days. With our open API, we can ensure everything is properly linked to your SCM and defect tracking system, then tailored to your specific needs.

Because Coverity integrates into your development process and assigns defects to the proper owners, we deliver a solution that helps you ensure defects are corrected while measuring the progress you make in improving the overall quality of your software.

Key Requirements for Effective Static Analysis

Coverity developed the first solution capable of automatically identifying defects with a combination of precision, speed, and accuracy that remains the standard for excellence in static code analysis. Our engineers have tackled some of the most difficult problems that historically hampered source code analysis: build integration, compiler compatibility, high rate of false positives, and effective root-cause analysis.

After working closely with more than 450 customers, Coverity knows what organizations need to make static analysis work for their development teams:

Accuracy

Coverity Prevent uncovers critical defects with a minimal number of false positives and false negatives. Key features include:

- Authentic compilation of source files
- Breadth of checkers
- Path simulation engine
- 100% data value coverage
- Boolean satisfiability engine
- Dedicated false path pruning

Scalability

Coverity Prevent scales to millions of lines of code while taking hours to run, not days or weeks. Key features include:

- Multi-core support
- Incremental analysis
- Non-redundant path analysis

Integration

Coverity supports all popular compilers and provides an open API so it can be configured to your unique build environment. Key features include:

- Automated build integration
- Extensive compiler support
- Fully scriptable
- Open standard interface support
- GUI and IDE support

Administration

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

- No changes to code or build scripts
- LDAP integration
- Customizable status and severity
- Assignment of defect ownership
- Email notification
- Customizable reporting

Sample Problems Coverity Detects

Crash-Causing Defects

- Null pointer de-reference
- Use after free
- Double free

Concurrency

- Deadlocks
- Lock Contention
- Race Conditions

Incorrect Program Behavior

- Dead code caused by logical errors
- Uninitialized variables
- Erroneous switch cases

Performance Degradation

- Memory leaks
- File handle leaks
- Custom memory and network resource leaks
- Database connection leaks
- Mismatched array new/delete
- Missing destructor

Improper Use of APIs

- STL usage errors
- API error handling
- API ordering checks

Security Vulnerabilities

- Array and buffer overrun
- Unsafe uses of tainted data

These defects are found with

- High accuracy; false positive rates are often below 20%
- 100% path coverage
- 100% value coverage
- Cross-module, cross-function analysis to identify complex errors involving multiple components

About Coverity

Coverity (www.coverity.com), the leader in improving software quality and security, is a privately held company headquartered in San Francisco. Coverity's groundbreaking technology enables developers to control complexity in the development process by automatically finding and helping to repair critical software defects and security vulnerabilities throughout the application lifecycle. More than 450 leading companies including ARM, Phillips, RIM, Rockwell-Collins, Samsung and UBS rely on Coverity to help them ensure the delivery of superior software.

Free Trial

Request a free Coverity trial and see first hand how to rapidly detect and remediate serious defects and vulnerabilities. No changes to your code are necessary. There are no limitations on code size, and you will receive a complimentary report detailing actionable analysis results. Register for the on-site evaluation at www.coverity.com or call us at (800) 873-8193.

Headquarters

185 Berry Street, Suite 2400
San Francisco, CA 94107
(800) 873-8193
<http://www.coverity.com>
sales@coverity.com

Boston

230 Congress Street, Suite 303
Boston, MA 02110
(617) 933-6500

UK

Coverity Limited
Magdalen Centre
Robert Robinson Avenue
The Oxford Science Park
Oxford OX4 4GA
England

Japan

Coverity Asia Pacific
Level 32, Shinjuku Nomura Bldg.
1-26-2 Nishi-Shinjuku, Shinjuku-ku
Tokyo 163-0532
Japan