coverity®

WHITE PAPER

Development Testing For Java Applications

More. Better. Faster.

Today's software development teams are under immense pressure to meet aggressive functionality, security, efficiency, quality and compliance goals. Users demand new releases faster, with top-notch quality and security. Considering the high cost of quality and security failures, it is more important than ever to address this risk throughout the software development process. Potential problems need to be spotted early in order to prevent release delays or, even worse, post-production failures. In today's economy, teams also need to demonstrate that they are creating value for their employer. They need visibility into software defects and advance warning of impending problems to deliver robust products on schedule and to prove the efficiency and performance of the team. In short, they need to deliver more features, better quality and security, faster to the market.

While these challenges exist for all development teams, regardless of market vertical or development language, this document will focus on Java because industry experts believe that there are more than 9 million Java developers. With user count in the billions, Java is one of the most widely used platforms. Moreover, Java is a modern language with some nice features to improve quality, security and developer productivity.

- The Java virtual machine uses a garbage collector to ensure objects are cleaned up automatically, which reduces the burden of developers managing resources.
- Java provides language constructs for concurrency and synchronization, enabling developers to more easily take advantage of concurrency.
- Java offers a security manager and sandboxes to limit the impact of vulnerabilities.

Java Developers Aren't Immune to Risk

While these features simplify development of many straightforward applications, they can also give developers a false sense of comfort in large, sophisticated projects. The garbage collector is unpredictable, so loosely managed resources can lead to resource exhaustion in addition to creating performance concerns. While the language understands concurrency, developers still need to be careful to avoid deadlocks and race conditions as complexity increases. The security manager will enforce the security policy, but developers need to carefully define and implement the policy to ensure proper protection. Even so, it is possible that platform defects will allow malicious code to escape the security manager. In fact, there were widely publicized vulnerabilities in the Oracle JVM in August 2012 and January 2013 that allowed attackers to successfully bypass the Java sandbox and install malware on the host system.

Even worse, these features provide little to no protection from application-level vulnerabilities like SQL injection and cross-site scripting, which account for 75% of reported attacks and can be exploited to access business-critical information. LinkedIn experienced such a breach in June 2012 that exposed the passwords of more than 6.5 million LinkedIn users and caused significant brand damage. That breach was attributed to an application-level vulnerability along with ineffective security planning.

Open Source Solutions Help Reduce Risk

Fortunately, there are a number of tools in the Java world to help address these gaps. Some of these are commercial solutions, and there are many high-quality free open source tools as well—with professional support available for both. These tools can help you understand the effectiveness of your development effort by identifying coding errors, enforcing coding standards, providing test frameworks and providing information about which code is executed. They are accessible to developers, easy to use within many development environments and very popular.

Unfortunately, it's not always obvious how certain types of users can take advantage of these tools. For example, many of these tools do a good job with their intended task and are easy to use with small teams or individual developers. Large-scale projects, and environments where access to information and code needs to be controlled, may not be able to use these tools in the traditional way. The good news is that the Java ecosystem provides a solution to this problem as well. There are enterprise-class development testing platforms that make these same tools accessible even to secure, global, distributed development organizations. Such solutions can actually add significant value for all organizations, by tying all information together and increasing visibility and efficiency.

These tools generally focus on specific aspects of software development, and getting a comprehensive view of your development effort often requires finding a combination of multiple tools that best meets the needs of your organization. That includes not just the information they can uncover, but how they can work together to help you understand the larger trends and take action to effectively manage your effort.

Designing Your Process

A robust development process is not the result of simply installing tools recommended by somebody else. There are a number of important considerations when selecting tools to be a part of your process. You need to understand the development challenges your team faces, including business requirements, external influences and individual expertise. You need to determine which tools are going to support your process and keep developers focused and productive. You need to consider the costs to acquire, deploy, run and maintain the tools, as well as the effort required to train users, evangelize adoption, and understand and manage the information generated by the tools.

Your development process needs to help you manage risk in your development effort. To ensure adoption, the tools should be easy for developers to use, they should provide obvious value to users and they should be tightly integrated into a consistent developer workflow. Two areas of particular concern are consistency and accuracy. Consistency is important because you need to minimize time spent interacting with different tools—that not only wastes

time, but increases training and maintenance costs, introduces opportunities for user error and can give the process a reputation for poor efficiency. This concern is often addressed by automating usage of the low-level tools and consolidating the data collected by the individual tools into a platform that helps organize issues and identify trends. Accuracy is important because you need to minimize time spent investigating bogus issues—as with consistency, these bogus issues have a direct impact on productivity and indirect impact on user sentiment.

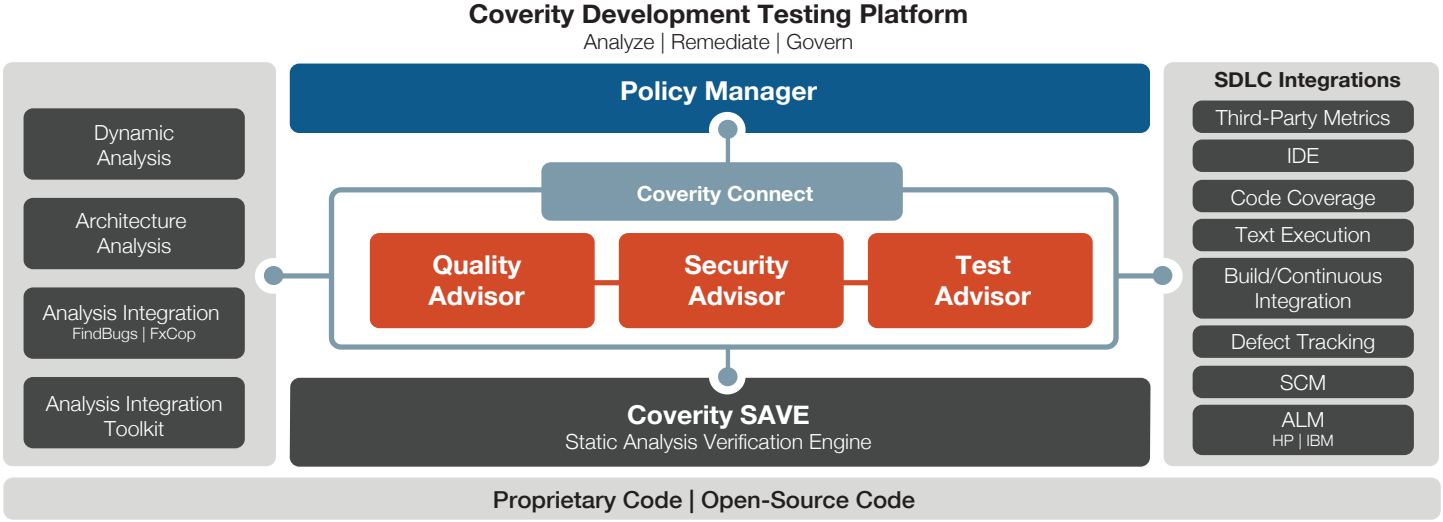
A related consideration is that the tools should provide the most relevant and specific information possible. Some analysis tools look for coding defects like null dereferences while others look for violations of coding rules like, "a class that implements Serializable must define a serialVersionUID field"; the tools you choose need to focus on the issues that you care about. If you don't care about enforcing coding standards, then you don't want to clutter your analysis results with coding standard violations. Similarly, it's good to know that a particular problem occurs at a particular place in the code, but it's better for the tools to tell you which variable is affected, what has happened to that variable prior to the problem, which code paths are affected by the problem and so forth.

There are considerations from Management's perspective as well: the process needs to (among other things) help avoid duplicated or wasted effort and ensure accountability. For example, you need a way for issues to be assigned an owner—the person responsible for moving that issue into the next stage of its lifecycle. It might be as simple as assigning coding issues to the person that most recently modified the related source code, but many teams need to do more. Perhaps a different team (like QA or Security) needs to sign off on the fix before it can be marked "Closed," or maybe the code author is not available to work on the issue. Low-level data from the tools needs to be interpreted or filtered before deciding what action to take. For example, testing and coverage tools identify test violations and coverage while developers need to know specifically which section(s) of code require additional tests—after filtering out code that is already tested, isn't important, can't be tested or is not actually used. If you have compliance or regulatory concerns, you also need to consider how usage of the tools can be monitored, results can be archived and whether sufficient access controls can be put into place.

For your process to be successful, it needs to provide value that clearly outweighs the cost of use. Developers and managers have to decide whether there is a net benefit in adopting the new process. It needs to enable your team to easily understand what to do: identify the problems that you care about, minimize the effort wasted jumping through hoops and sifting through the noise and boil problems down to specific action items with clear owners. Ultimately, the success of your team—and the process—will be measured by your ability to meet your obligations and deliver quality product on schedule. An unused tool is, of course, worthless; a process built around an unused tool is even worse. How can you rely on a process that isn't consistently used?

Enter Coverity

The Coverity® Development Testing Platform is designed to tie disparate parts of your software development lifecycle (SDLC) together, acting as a hub for team members from developers to executives, keeping them focused on their quality and security goals and giving them visibility into the development process.

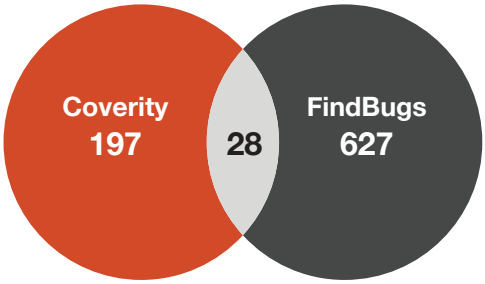


It is common for analysis tools to identify more issues than can be realistically addressed. In order to wisely manage resources, teams must be able to effectively manage, filter and prioritize those issues. For compliance, it is important to demonstrate the effectiveness and adoption of your quality, security and risk management processes. When there are problems in the field, it is important to recognize past mistakes so they can be avoided in the future.

Each team's development process probably uses a few different, specialized analysis tools to find important coding problems. The Coverity SAVE® Static Analysis Verification Engine is designed to complement common analysis tools by finding the important defects that those tools often miss. It is highly tuned to reduce noise and highly configurable to adapt to your coding idioms and standards. It does extremely sophisticated analysis of complex, inter-procedural data flows, utilizing patented techniques to automatically filter out false positives while finding real, important defects. This keeps the list of issues manageable while making it significantly more likely that each issue is something that you care about. It has proven itself on code bases ranging from thousands to millions of lines of code, from teams of a handful of developers to thousands. Coverity Quality Advisor adds specific analysis rules on top of SAVE to find the most critical quality issues.

Coverity SAVE and Coverity Quality Advisor focus on the problems that not only have a significant impact in your project, but that are extremely difficult to identify through other means. For example, a null dereference that affects just one code path that spans multiple files and functions would be a nightmare to track down through code review or even with a debugger. A race condition caused by a single, unsynchronized method accessing a data member in an otherwise safe class would be extremely time consuming, if not practically impossible to find. Coverity SAVE truly understands how the pieces fit together, enabling sophisticated algorithms that accurately find problems such as these in a reasonable amount of time.

To demonstrate this difference, Coverity analyzed the source code for version 1.496 of the Jenkins job management system often used for automated software builds (http://www.jenkins-ci.org) using release 6.5.1 of the Coverity Development Testing Platform and FindBugs™ version 2.0.1, with all checks enabled. Of 852 unique issues identified, only 28 issues were identified by both Coverity and FindBugs. Coverity found 197 unique issues, with 188 of those coming from high-impact categories (security and

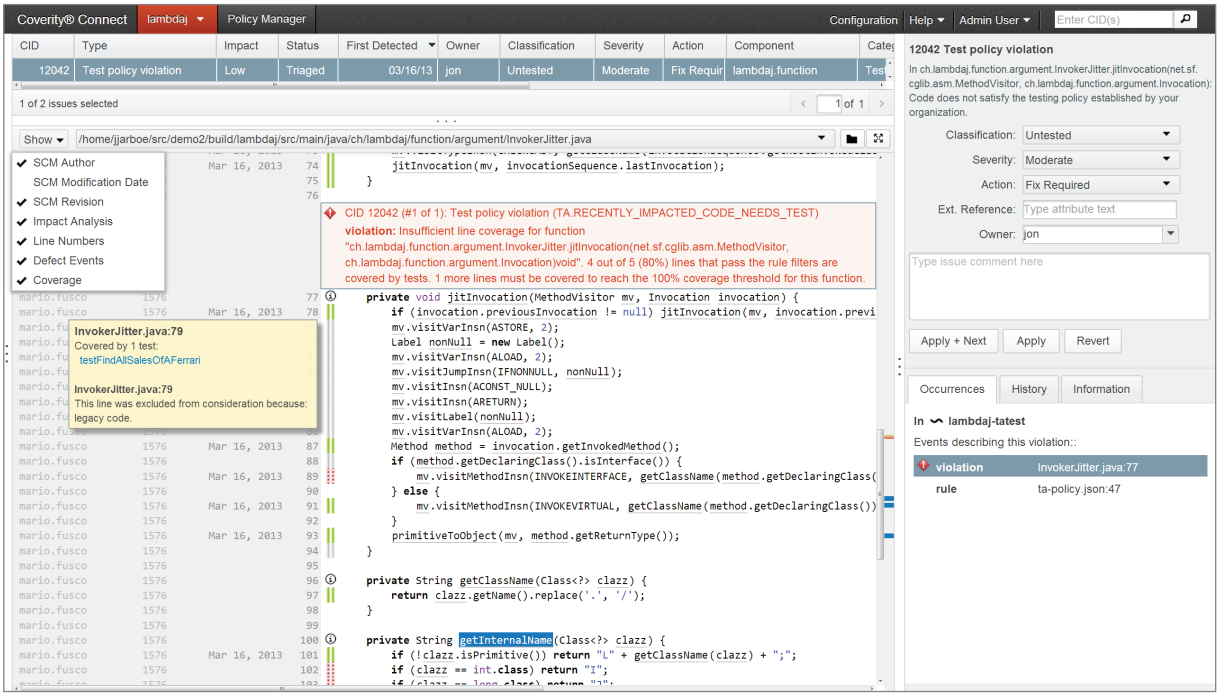


concurrency bugs, resource leaks and unhandled exceptions like null dereferences). FindBugs found 627 unique issues, with 29 coming from those high-impact categories. Both found interesting unique issues, and your needs will dictate whether you should use either or both as a part of your process.

Coverity Dynamic Analysis and Coverity Architecture Analysis look for problems from a different perspective. Many of the most difficult problems to find, such as resource leaks and concurrency problems, are the result of complex runtime relationships that developers didn't anticipate or fully understand. Coverity Dynamic Analysis is especially designed to quickly find exactly these problems by watching your program as it runs. Coverity Architecture Analysis helps you understand and manage the complex relationships between parts of your code so you can enforce modularity and interfaces to reduce the opportunity for problems.

The Coverity platform and Coverity Security Advisor understand web application frameworks like Spring MVC. This enables our analysis to accurately follow control and data flow even when it's expressed via annotations, framework configuration files or container convention as opposed to traditional Java source code. Coupled with Coverity SAVE's detailed understanding of program behavior, we're able to accurately report security problems before your application goes to audit. Unlike most security audits, the results are conveyed in a developer-friendly manner, including the problem, the recommended fix and the location where the fix should be implemented—which is often not where the problem manifests.

Coverity Quality Advisor and Coverity Security Advisor surface all of these quality and security issues, as well as defects identified by other analysis tools—even homegrown, open-source and proprietary ones within Coverity Connect. The import process is configured to normalize issues from different analysis engines ensuring consistency and helping developers easily prioritize different types of issues as the cream rises to the top.



Automatically identify missing unit tests in critical areas of the code.

Coverity Test Advisor improves your testing efficiency and effectiveness by identifying new test cases that will help make your product more robust and merging unit test and coverage information with analysis data from other parts of the platform, such as Coverity SAVE. In addition to helping you manage test results by identifying failed tests, it also recommends where missing tests should be added. To keep users focused on the important areas, it provides sophisticated testing coverage rules for important code based not only on file and function, but on code constructs, when code was last impacted by code changes and other criteria. Your testing success doesn't depend on manual evaluation of low-level data—you tell the solution how to find important code, and it will automatically notify you when the important code is not being adequately tested, even recommending which test suite or test case you might want to extend. Developers are automatically focusing on the important code when they address these issues.

Coverity Connect is the unified issue management console for the Coverity Development Testing Platform. The platform consolidates all issues in a single place, making it easy for developers to use Coverity Connect to find and prioritize the issues that need to be addressed as part of a consistent workflow. Regardless of source, all issues can be assigned an owner so there is clear ownership and accountability, avoiding wasted effort as well as issues falling through the cracks. The platform tracks issues across branches and builds, so you can easily understand the impact of issues in global and local contexts. It can integrate with the most popular IDEs, so developers can work in a familiar, productive environment. It supports the enforcement of enterprise-grade access controls and audit trails, so you can rest assured that your code, data and artifacts are safe and secure. Automation ensures that developers remain productive and accurate data flows to the relevant people and processes and management has visibility into the state of development.

Users gain these benefits for all issues managed in the Coverity platform, even those from third-party tools like FindBugs or homegrown solutions that don't have their own sophisticated issue-management capabilities. In fact, the Coverity platform includes a preconfigured integration with FindBugs to reduce noise while taking advantage of Coverity's sophisticated analysis and defect-management capabilities.

Coverity Policy Manager enables management to track the important metrics and understand at a glance whether development is proceeding appropriately. Policy violations are clearly identified, with the ability to drill down and find the source of those problems so they can be fixed. Because the data is consolidated, normalized and fresh, management can easily spot emerging problems. It is easy to demonstrate compliance with

development policy, adopt the platform, and prove efficiency and performance. Even better, products are released to QA and Security Audit with fewer defects, leading to faster, more predictable release schedules.

As an example, consider a team whose process utilizes the Coverity Development Testing Platform, FindBugs, JUnit and Cobertura. They have tuned, highly accurate static analysis in the form of Coverity SAVE and FindBugs enables them to find a broad range of coding and security errors while code is developed; Coverity Dynamic Analysis to identify critical problems that can only be found at run time; JUnit as a testing framework to verify that the application behaves as intended; Cobertura to understand which parts of the application are being tested; and Coverity Test Advisor to keep developers focused on testing the important code. All the while, Coverity Connect ties these process components into a unified, enterprise-grade platform that enables developers to quickly and easily prioritize, manage and fix all identified issues. Coverity Policy Manager enables Management's governance, risk management and compliance efforts. If a need arises to detect additional problems with other tools or custom checkers, the platform will easily accommodate that while maintaining the same process, workflow and capabilities.

Why Coverity

The Coverity Development Testing Platform allows Java developers to accurately identify and easily remediate many classes of high-impact software problems such as:

- Web security defects like injection and cross-site scripting
- Memory and resource leaks
- Deadlock and other concurrency problems
- Improper or unsafe use of null references
- Incorrect use of APIs like Android, JPA, Hibernate and Spring
- Performance and maintainability issues
- Testing policy violations

In addition to providing world-class analysis capabilities—with accuracy and relevance recognized as second to none—it unifies all of the tools you use as part of your development process, enabling developers to find, manage and resolve software defects easily and quickly.

Management gains visibility into the state of development, with metrics that identify risk so it can be addressed early, and that demonstrate effective management throughout the development lifecycle. More importantly, real issues are addressed early in the development lifecycle leading to faster, more predictable releases.

Coverity, the leader in development testing, is the trusted standard for companies that need to protect their brands and bottom lines from software failures. More than 1,100 Coverity customers use Coverity's development testing platform to automatically test source code for software defects that could lead to product crashes, unexpected behavior, security breaches or catastrophic failure. Coverity is a privately held company headquartered in San Francisco. Coverity is funded by Foundation Capital and Benchmark Capital.

For More Information

Find out how Coverity can help your organization improve the quality and security of your Java code. To learn more, contact your Coverity representative or visit us at www.coverity.com

coverity®

For More Information
www.coverity.com
Email: info@coverity.com

Coverity Inc. Headquarters
185 Berry Street, Suite 6500
San Francisco, CA 94107 USA

U.S. Sales: (800) 873-8193
International Sales: +1 (415) 321-5237
Email: sales@coverity.com

© 2013 Coverity, Inc. All rights reserved. Coverity, the Coverity logo and Coverity SAVE are registered trademarks of Coverity, Inc. in the U.S. and other countries. All other company and product names are the property of their respective owners.