

Automatically Identify and Prevent Java™ Errors throughout the Entire Development Lifecycle with Parasoft® Jtest®

Parasoft Jtest is a comprehensive Java testing product for development teams building Java EE, SOA, Web, and other Java applications. Whether a team is trying to build quality into new code or extend a legacy code base without breaking existing functionality, Jtest provides them a practical way to ensure that their Java code works as expected. It empowers them to modify their code quickly and with confidence, resulting in optimization of development resources and control of development schedules and costs.

Automatically Expose Difficult-to-Find Runtime Bugs

BugDetective is a new static analysis technology that searches a code base for errors which can lead to runtime bugs and application instabilities. By automatically tracing and simulating execution paths through even the most complex applications—those with paths that span multiple methods, classes, and/or packages and contain dozens of sequence calls—BugDetective exposes bugs that would be very difficult and time-consuming to find through manual testing or inspections, and would be exponentially more costly to fix if they were not detected until runtime. Using BugDetective, developers can find, diagnose, and fix classes of software errors that can evade coding standards analysis and/or unit testing. Exposing these bugs early in the software development lifecycle saves hours of diagnosis and potential rework.

Automate Code Review

To help developers identify small coding mistakes before they spawn bugs that are difficult to find and fix, Jtest's coding standards analysis automatically checks whether code follows over 700 Java coding best practices and any number of custom coding rules. Coding standards analysis can also be used to prevent application-specific errors, enforce a security policy, improve code readability and maintainability, and identify code that would benefit from refactoring. Many coding problems can be resolved automatically, so code can be improved in just seconds.

Since this automated coding standards analysis virtually eliminates the need for line-by-line inspections during peer code reviews, these reviews can focus on high-return value analysis, such as examining design, algorithmic, or implementation issues. Peer code reviews are further supported by Jtest's Code Review module, which automates the review process to facilitate participation and communication. This makes code reviews more productive and practical for software development organizations—especially those with distributed teams.

Subjecting code to automated and manual inspections ensures that quality is built into the code, which allows QA to focus on high-level verification, reduces time to market, and improves project predictability.

Verify and Capture Code Behavior at the Unit, Component, and Application Level

Developers and QA testers can use Jtest to build a test suite that not only verifies code correctness and reliability at multiple levels, but also captures code behavior to establish a baseline for regression testing.

Benefits

- **Modify existing code quickly, and with confidence** – Enables teams to quickly build a regression safety net that will expose defects immediately upon introduction and determine if code modifications break existing functionality – even if the team has a large existing code base with no tests or minimal tests.
- **Control development costs and schedules** – Exposes errors as early as possible, which is when they are fastest and cheapest to fix. Tests a broad range of potential user paths to uncover difficult-to-find problems that could delay releases or require post-release patches.
- **Optimize development resources** – Automatically vets approximately 80% of coding issues so developers can spend less time on line-by-line inspections and debugging, and more time on design, algorithms, and implementation.
- **Leverage the power of the latest technologies while controlling their risks** – Reduces the difficulty of testing complex enterprise applications (such as SOA/Web services and Java EE applications).
- **Gain instant visibility into Java code's quality and readiness** – Provides on-demand objective code assessments and tracks progress towards quality and schedule targets.

Features

- Automatically creates sensitive low-noise regression test suites—even for large code bases.
- Automatically finds runtime bugs in execution paths that may cross multiple methods, classes, or packages.
- Generates functional JUnit test cases that capture actual code behavior as a deployed application is exercised.
- Generates extendable JUnit and Cactus (in-container) tests that expose reliability problems and capture behavior.
- Executes the test suite to identify regressions and unexpected side effects.
- Monitors test coverage and achieves high coverage using branch coverage analysis.
- Identifies memory leaks during test execution.
- Checks compliance to configurable sets of over 700 built-in rules, including 100 security rules.
- Corrects violations of 250 rules.
- Allows creation of custom rules by modifying parameters, using a graphical design tool, or providing code that demonstrates a sample rule violation.
- Calculates metrics.
- Identifies and refactors duplicate and unused code.
- Supports Struts, Spring, Hibernate, EJBs, JSPs, servlets, and so on.
- Full integration with Eclipse, RAD, JBuilder.
- Limited integration (result import only) with IntelliJ IDEA and Oracle JDeveloper.
- Integration with most popular source control systems.
- Automates the peer code review process (including preparations, notifications, and routing).
- Shares test settings and files team-wide or organization-wide.
- Generates HTML and XML reports.
- Provides GUI (interactive) and command-line (batch) mode.

System Requirements

Operating System

- Windows: Windows 2000, XP, 2003, or Vista
- Linux: Red Hat 9.0, Fedora Core 1-3 or higher, Red Hat E.L. 2, 3, 4
- Solaris: Solaris 8, 9, 10

Hardware

- Intel® Pentium® III 1.0 GHZ or higher recommended
- UltraSPARC processor 1.0 GHZ or higher recommended
- 1 GB RAM minimum; 2 GB RAM recommended
- Sun Microsystems JRE 1.3 or higher (32-bit)

IDE (for plugin only)

- Eclipse 3.3-3.0, IBM Rational Application Developer 7.0-6.0, JBuilder 2007

As soon as a Java method or class is implemented, the developer can test it in isolation to start finding errors instantly. To facilitate this initial developer testing, Jtest automatically generates extendable, high-coverage JUnit tests that expose reliability problems and can be leveraged for regular regression testing.

Once Java EE classes are deployed to a local application server, Jtest can automatically generate Cactus tests for them and then execute those tests in the application container to simulate the code's realistic runtime environment. This ability to test complex, difficult-to-test Java EE applications in isolation (on the desktop or a local server) before they are deployed to a production system promotes early exposure of defects that are typically not noticed until QA or later—when finding and fixing them is significantly more difficult and expensive.

By monitoring a deployed application in real time, Jtest Tracer can capture realistic functional tests to further extend the regression test suite. Simply use the application's GUI or a test client (such as Parasoft SOAtest for SOA/Web services or Parasoft WebKing for Web applications) to execute the use cases you want to verify, then Jtest Tracer will capture these operations in "positive" JUnit test cases. If the functionality associated with your use cases later breaks, these test cases will fail.

Establish an Automated Infrastructure for Regular Automated Regression Testing

Collectively, these test cases establish a robust regression test suite that automatically runs on a regular basis to detect defects immediately upon introduction and determine if code modifications break existing functionality. Having such a regression test suite helps developers rapidly change code with confidence, which is especially critical for teams working on complex and constantly-evolving applications. Even if the team has a large existing code base with no tests or minimal tests, they can use Jtest to create a robust, low-noise regression suite overnight.

Integrate Jtest into the Team's Infrastructure and Workflow

Jtest's support for team deployment standardizes testing team-wide and provides a sustainable workflow for integrating best practices into the team's existing processes—with minimal disruption. The architect defines the team's designated test configurations, then Parasoft Team Configuration Manager (TCM) automatically shares them across all team Jtest installations. Developers can test code directly from their IDE to find and fix problems before adding it to source control. Additionally, Jtest Server can test the entire project code base each evening, then email reports to the manager and responsible developers if any problems are detected. Developers can then import results into their IDEs to review and repair the errors reported for code they authored. Jtest Server also sends information from these tests to the Parasoft Group Reporting System (GRS), which collects and analyzes data from Jtest and other testing products, and then organizes data into role-based dashboards that provide managers, architects, developers, and testers instant visibility into the project's overall quality and status.

www.parasoft.com

Contact info:

Parasoft Corporation, 101 E. Huntington Dr., 2nd Flr., Monrovia, CA 91016

Ph: (888)305.0041, Fax: (626)256.6884, Email: info@parasoft.com