PARASOFT Jtest®

DATA SHEET

Parasoft® Jtest® is an integrated solution for automating a broad range of practices proven to improve development team productivity and software quality. It focuses on practices for validating Java code and applications, and it seamlessly integrates with Parasoft SOAtest to enable end-to-end functional and load testing of today's complex, distributed applications and transactions.

Parasoft's customers, including the majority of the Fortune 500, rely on Jtest for:

- Preventing defects that impact application security, reliability, and performance
- Complying with internal or regulatory quality initiatives
- Ensuring consistency across large and distributed teams
- Increasing productivity by automating tedious yet critical defect-prevention practices
- Successfully implementing popular development methods like TDD, Agile, and XP

Capabilities

Static code analysis	Facilitates regulatory compliance (FDA, PCI, etc.). Ensures that the code meets uniform expectations around security, reliability, performance, and maintainability. Eliminates entire classes of programming errors by establishing preventive coding conventions.
Data flow static analysis	Detects complex runtime errors related to resource leaks, exceptions, SQL injections, and other security vulnerabilities without requiring test cases or application execution.
Metrics analysis	Identifies complex code, which is historically more error-prone and difficult to maintain.
Peer code review process automation	Automates and manages the peer code review workflow- including preparation, notification, and tracking- and reduces overhead by enabling remote code review on the desktop.
Unit test generation and execution	Enables the team to start verifying reliability and functionality before the complete system is ready, reducing the length and cost of downstream processes such as debugging.
Runtime error detection	Automatically exposes defects that occur as the application is exercised-including race conditions, exceptions, resource & memory leaks, and security attack vulnerabilities.
Test case "tracing"	Generates unit test cases that capture actual code behavior as an application is exercised providing a fast and easy way to create the realistic test cases required for functional/regression testing.
Automated regression testing	Generates and executes regression test cases to detect if incremental code changes break existing functionality or impact application behavior.
Coverage analysis	Assesses test suite efficacy and completeness using a multi-metric test coverage analyzer. This helps demonstrate compliance with test and validation requirements such as FDA.
Team deployment and workflow	Establishes a sustainable process that ensures software verification tasks are ingrained into the team's existing workflow and automated so team members can focus on tasks that truly require human intelligence.

These core capabilities are also available for C, C++, .NET languages.

Error assignment and distribution	Facilitates error review and correction. Each issue detected is prioritized, assigned to the developer who wrote the related code, and distributed to his or her IDE with direct links to the problematic code.
Centralized reporting	Ensures real-time visibility into quality status and processes. This helps managers assess and document trends, as well as determine if additional actions are needed for regulatory compliance.
Continuous "On-the-fly" static analysis	Automatically run static analysis in the background as developers review, add, and modify code. This helps the team identify and fix problems as soon as they are introduced.

Key Features

- Built-in support for Google Android, Spring, Hibernate, Eclipse plug-ins, TDD, JSF, Struts, JDBC, EJBs, JSPs, servlets, and more (mobile, embedded, Java EE...)
- Integrates with Parasoft SOAtest for end-to-end functional and load testing for web, SOA, and cloud development.
- Exposes runtime defects that occur as the application is exercised by unit, manual, or scripted tests–including race conditions, exceptions, resource leaks, and security attack vulnerabilities
- Without requiring execution, identifies execution paths that can trigger runtime defects
- Checks compliance to configurable sets of over 1000 built-in static analysis rules for Java
- Provides templates for OWASP Top 10, CWE-SANS Top 25, PCI DSS, and other security static standards
- Automatically corrects violations of 350+ rules with QuickFix
- Allows easy GUI-based customization of built-in rules
- Identifies and prevents concurrency defects such as deadlocks, race conditions, missed notification, infinite loops, data corruption other threading problems
- Automatically creates robust low-noise regression test suites–even for large code bases
- 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 achieve high coverage using branch coverage analysis
- Integrates and extends manually-written unit test cases
- Continuously executes the test suite to identify regressions and unexpected side effects
- Performs runtime error detection as tests execute
- Parameterizes test cases for use with varied, controlled test input values (runtime-generated, user-defined, or from data sources)
- Monitors test coverage with multiple metrics
- Tracks code coverage from manual tests and test scripts
- Steps through tests with the debugger
- Tests individual methods, classes, or large, complex applications
- Calculates metrics such as Inheritance Depth, Lack Of Cohesion, Cyclomatic Complexity, Nested Blocks Depth, Number Of Children
- Identifies and refactors duplicate and unused code
- Automates the peer code review process (including preparations, notifications, and routing)
- Shares test settings and files team-wide or organization-wide
- Generates HTML, PDF, XML, and custom reports
- Tracks how test results and code quality change over time
- Provides GUI (interactive) and command-line (batch) mode

Infrastructure Support

- Full integration with Eclipse 3.2-3.7, IBM Rational Application Developer 7.0-8.0
- Integration with Ant, Maven, CruiseControl, Hudson, and other build & release tools
- Integration with most popular source control systems
- Open Source Control API, which allows teams to integrate any other source control system

System Requirements

Operating System

- Windows: 7, Vista, 2000, XP, or 2003 (x86 or x86_64)
- Linux: Red Hat E.L. 3, 4, 5 or equivalent (x86 or x86_64)
- Solaris: Solaris 10 (SPARC)
- Mac: OS X 10.5 or higher

Hardware

- Intel® Pentium® III 1.0 GHZ or higher recommended
- 512 MB RAM minimum; 2 GB RAM recommended
- JRE 1.3 or higher

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