Parasoft C++test for Visual Studio

Parasoft C++test is an integrated solution for automating a broad range of best practices proven to improve software development team productivity and software quality. C++test enables coding policy enforcement, static analysis, comprehensive code review, unit and component testing, and runtime error detection to provide teams a practical way to ensure that their C and C++ code works as expected. C++test can be used both on the desktop under leading IDEs as well as in batch processes via command line interface for regression testing. C++test integrates with Parasoft Development Testing Platform, which provides interactive Web-based dashboards with drill-down capability, allowing teams to track project status and trends based on C++test results and other key process metrics.

C++test can help development teams:

- Apply a comprehensive set of best practices for identifying and addressing defects from the earliest phases of the development cycle—when fixing them requires minimal effort and rework.
- Automatically vet known coding issues so more time can be dedicated to tasks that require human intelligence.
- Efficiently construct, continuously execute, and maintain a comprehensive regression test suite that detects whether updates break existing functionality.
- Gain instant visibility into C and C++ code quality and readiness by accessing on-demand objective code assessments and tracking progress towards quality and schedule targets.
- Build an efficient, consistent, and controlled team workflow for applying best practices that reduce testing time, testing effort, and the number of defects that reach QA.
- Automate negative testing on a broad range of potential user paths to uncover problems that might otherwise surface only in “real-world” usage.

Available C++test 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. For details, see Static Code Analysis.

Data Flow Analysis

Detects complex runtime errors related to resource leaks, exceptions, SQL injections, and other security vulnerabilities without requiring test cases or application execution. For details, see Flow Analysis.

Metrics Analysis

Identifies complex code, which is historically more error-prone and difficult to maintain. For details, see Metrics Calculation, Static Code Analysis.

Unit Test Generation and Execution (Including Coverage & Regression Testing)

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. For details, see Test Creation and Execution.

Runtime Error Detection

Exposes critical defects (such as race conditions, exceptions, resource leaks, and security attack vulnerabilities) as the application is exercised during unit testing or at the application level. For details, see Runtime Error Detection.

Peer Code Review

Automates and manages the peer code review workflow- including preparation, notification, and tracking- and reduces overhead by enabling remote code review on the desktop. For details, see Code Review.

Change-Based Testing

Parasoft's change-based testing helps you optimize your testing efforts by identifying and executing only the test cases directly related to your most recent source code modifications. Not having to test the entire system after each modification yields tremendous productivity improvements. For details, see Change-Based Testing and Traceability.

Requirement and Defects Traceability

In addition to graphically reporting requirement and defect status as indicated by developers, Parasoft DTP traces requirements and defects back to test cases. At a glance, the team can gain an objective assessment of which requirements are actually working as expected, which defects are resolved, and which still requirements and defects need testing. This real-time visibility into true requirement and defect status helps you prevent late-cycle surprises that threaten to derail schedules and budgets. For details, see Change-Based Testing and Traceability.

C++test Embedded Documentation

For general embedded testing documentation, see Cross-Platform and Embedded Testing.

For environment-specific information, see Supported Environments Details.