

DATA SHEET

Automate .NET Development with Parasoft® .TEST™

Increase software development team productivity and software quality

Parasoft® .TEST™ is an integrated solution for automating a broad range of best practices proven to increase software development team productivity and software quality. Parasoft .TEST ensures developers that their .NET code works as expected by enabling coding policy enforcement, static analysis, and unit testing. Parasoft .TEST also saves development teams time by providing a streamlined manual code review process. Parasoft .TEST can be used both on the desktop as a Microsoft Visual Studio plugin and in batch processes via command line interface for regression testing.

Parasoft .TEST works with programming languages that target the Microsoft .NET Framework, including C#, VB.NET, and MC++ (Managed C++). It can test any file or assembly that has been built to take advantage of the .NET CLR.

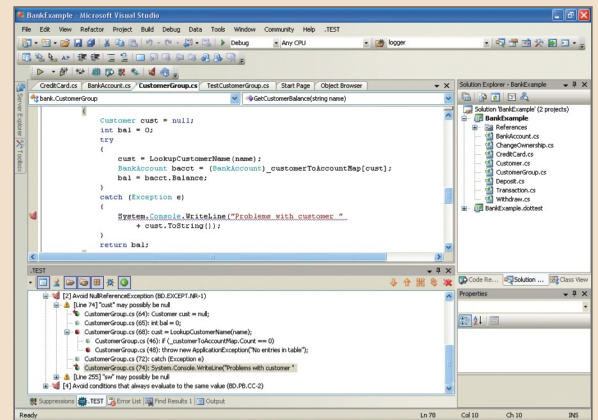
Identify Runtime Bugs without Executing Software

BugDetective is a new type of static analysis technology that uses several analysis techniques, including simulation of application execution paths, to identify paths that could trigger runtime defects. Defects detected include NullReferenceExceptions, resource leaks, SQL injections, and other security vulnerabilities. Such defects may also point to loopholes in the design of the code for the specific use cases corresponding to highlighted execution paths.

BugDetective's ability to expose bugs without executing code is especially valuable for users with legacy code bases lacking robust test suites – where runtime analysis and detection of such errors is not effective or possible. Reviewing the violations reported by BugDetective often leads to good coding conventions that can prevent future occurrences of such defects.

Automate Code Analysis for Compliance

A properly implemented coding policy can eliminate entire classes of programming errors by establishing preventive coding conventions. .TEST statically analyzes code to check compliance with such a policy. To configure .TEST to enforce a coding standards policy specific to their group or organization, users can define their own rule sets with built-in and custom rules. .TEST includes 200+ rules that check compliance with Microsoft's ".NET Framework Design Guidelines." Additional sets of rules cover CLS Compliance, Object Oriented Metrics, and Security. In addition to rules that examine the IL code, .TEST also provides rules that examine the C# source code; this enables .TEST to check for many code issues that cannot be identified by IL-level analysis (for example, formatting issues, empty blocks, misuse of operators, etc.). Custom IL-level and C# rules, which are created with a graphical RuleWizard editor, can also enforce specific project and organizational requirements and prevent the recurrence of application-specific defects after a single instance has been found.



.TEST's BugDetective identifies critical bugs without executing the code.

Benefits

- **Increase team development productivity** – Apply a comprehensive set of best practices that reduce testing time, testing effort, and the number of defects that reach QA.
- **Achieve more with existing development resources** – Automatically vet known coding issues so more time can be dedicated to tasks that require human intelligence.
- **Build on the code base with confidence** – Efficiently construct, continuously execute, and maintain a comprehensive regression test suite that detects whether updates break existing functionality.
- **Gain instant visibility into code quality and readiness** – Access on-demand objective code assessments and track progress towards quality and schedule targets.
- **Reduce support costs** – Automate negative testing on a broad range of potential user paths to uncover problems that might otherwise surface only in "real-world" usage.

Features

- Static analysis of code for compliance with user-selected coding standards
- Graphical RuleWizard editor for creating custom coding rules
- Static code path simulation for identifying potential runtime errors
- Streamlined code review process with a graphical interface and progress tracking
- Automated generation and execution of unit tests
- Flexible stub framework for use in unit tests
- Full support for regression testing
- Code coverage analysis with code highlighting
- Full team deployment infrastructure for desktop and command line usage
- Seamless integration with Microsoft Visual Studio .NET

Platforms

- Windows XP or Windows 2003 server

Compilers

- Visual Studio .NET 2005 or 2003

Since the rules in .TEST come from several sources, including the experiences of software developers in many companies, these rules help you learn from your own mistakes as well as the mistakes of others. This type of static analysis virtually eliminates the need for line-by-line inspections during peer code reviews. Reviews can then focus on examining algorithms, reviewing design, and searching for subtle errors that automatic tools cannot detect. While unit testing can identify issues with your code, automatic code analysis is the only way to prevent entire classes of errors from occurring in your code.

Enable Effective and Comprehensive Team Code Review

The innovative Code Review module, which automates preparation, notification, and tracking of peer code reviews, addresses the known shortcomings of this very powerful development practice. .TEST automatically identifies updated code by scanning the source control system, matches the code with designated reviewers, and tracks the progress of each review item until closure. With the Code Review module, teams can establish a bulletproof review process – where all new code gets reviewed and all identified issues are resolved.

Automate Unit and Component Testing for Instant Verification and Regression Testing

.TEST's automated testing helps establish the correctness and reliability of newly developed or legacy code. .TEST automatically generates complete NUnit tests, including test drivers and test cases, for any .NET language file or assembly. By using corner case conditions, these automatically generated test cases check function responses to unexpected inputs, exposing potential reliability problems. To verify functional correctness of code, additional tests can be added by extending the generated test cases. A coverage analyzer helps users gauge test suite efficacy and completeness, and demonstrate compliance with test and validation requirements. These automated testing capabilities are especially helpful for supporting automated continuous integration and testing as well as "test as you go" development.

.TEST provides the following features to facilitate unit testing:

- Quick creation of a baseline regression suite, which can be extended manually if necessary.
- A simple way to create assertions automatically while writing tests manually. This can save significant amounts of time, allowing developers to focus on creating more tests.
- Stub support, which allows classes to be tested in isolation. This addresses one of the greatest challenges in writing unit tests: getting complex objects in different states.
- Coverage information for the unit tests (both automatically generated and manually written tests). This helps developers focus their manual test writing efforts on code that has not yet been covered.

Establish a Uniform Process Across the Team with a User-Friendly Workflow

An easy-to-use workflow and a good deployment strategy are both essential for making the above features work in real world development processes. .TEST'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 .TEST installations. Developers can test code directly from Visual Studio to find and fix problems before adding it to source control. Additionally, .TEST 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 Visual Studio to review and repair the errors reported for code they authored.

.TEST also sends information from these tests to Parasoft's GRS reporting system, which provides interactive Web-based dashboards with drill-down capability, allowing teams to track project status and trends based on .TEST results and other key process metrics.

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