



for IEC 61508
for ISO 26262

 PARASOFT®

C++test™

C/C++対応自動テストツール

Ver.9.0

TÜV SÜD社によるISO 26262 / IEC 61508ツール認証を取得

静的解析、単体テスト、フロー解析、実行時メモリエラー検出を自動実行
ソフトウェアの品質向上と効率的な開発の実現をサポート



- テストケース、スタブ、テストドライバを自動生成
- 単体テスト、実行時メモリエラー検出を自動実行
- 7種類のコードカバレッジを測定
- 実機(ターゲット機)で、単体テスト、実行時メモリエラー検出、コードカバレッジ測定を自動実行
- コーディング規約チェックでソースコードを検証
- 処理フローを解析してエラーを検出(バグ探偵)

TechMatrix

テスト実行を自動化し、ソフトウェアの品質向上に威力を発揮するC++test。

動的解析

単体テスト

テストドライバー、スタブ、テストケースを自動生成し、単体テストを自動実行します。7種類のコードカバレッジを測定。テストの妥当性を確認できます。

アプリケーション検証

単体テスト時、アプリケーション実行時にメモリ領域を監視し、実行時メモリエラーを検出します。ターゲット環境でもエラーの検出が可能です。

※ターゲット環境でエラー検出を実施する場合は、Cross Platform Editionのライセンスが必要です。

オール
インワン

静的解析

コーディング規約チェック

MISRAやJSF、Effective C++などを含む1480個のコーディングルールでソースコードを検証。クラッシュを引き起こす可能性のあるコードを検出します。

静的フロー解析(バグ探偵)

パスの処理フローを静的に検証し、メモリークやバッファオーバーフローを引き起こす可能性のあるコードを検出します。

※「バグ探偵」は、Server Editionのみに含まれる機能です。

動的解析

単体テスト

テストドライバー、スタブ、テストケースを自動生成、テストを自動実行

プログラムを最大限にカバーし、かつ、例外が発生しやすいテストドライバー、スタブ、テストケースを自動生成し、テストを自動実行します。自動生成テストケースには、関数の型の最小値/最大値、0近辺の値、空文字、null、ランダム値といった値に加えて、ソースコード中にある定数の境界値(条件文で指定されている定数とその境界値を含む周辺の値)も含まれます。自動生成テストケースにより、予期せぬ値が渡されたときの関数の振る舞いを確認できます。

自動生成されるテストケースの値や初期化の制御

テストケースウィザードで引数や事前条件、事後条件、戻り値といった値を関数ごとに設定したり、自動生成テストケースの値を設定できます。また、グローバル変数の初期化や、クラスまたは構造体のオブジェクトの初期化を制御できるので、より現実的な状況を反映したテストが可能です。

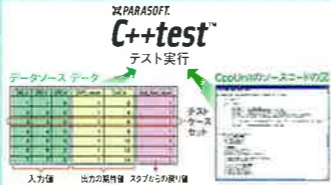
初期化・事前条件・事後条件の制御

- ファクトリ関数
グローバル変数やオブジェクトの初期化の際に返す型を設定し、初期化を制御するための関数です。ファクトリ関数は複数設定できるので、場面や状況の違いによるテスト対象の振る舞いを確認できます。
- privateメンバー変数の利用
テストケースの事前条件や事後条件として、privateフィールドの値を利用したり、オブジェクトの初期化でprivateフィールドを使用するよう設定できます。
- グローバル変数の初期化
自動生成テストケースに関連したグローバル変数を自動的に初期化するよう設定できます。



外部ファイルに格納されているテストケースやCppUnitテストケースのインポート

関数が参照している引数やグローバル変数、戻り値について、入力値やその期待値、および、スタブからの戻り値を、CSVやExcel、データベースなどの外部ファイルからインポートし、テストを実行できます。既存のCppUnitテストケースもインポートしてテストを実行できます。CppUnitテストケースを使用したテストでもテストカバレッジの測定が可能です。バージョンアップや再開発といった過去に作成したコードを利用する開発でC++testを導入した場合でも、既存の資産を有効活用できます。



スタブの設定を容易にするスタブビュー

アクセスできない外部リソースへのアクセスがある場合、代替となるスタブを自動生成してテストを実行します。自動生成されたスタブソースコードの編集、ユーザー定義スタブの追加/編集など、スタブに関する詳細な設定をスタブビューから実施することが可能です。

7種類のテストカバレッジ分析をレポート

テスト実行中に7種類のテストカバレッジを測定し、テスト終了後に行、関数、ファイル単位でレポートします。

C++test がレポートするカバレッジ分析

- 行カバレッジ
- ブロックカバレッジ
- ステートメントカバレッジ(C0:命令網羅率)
- バスカバレッジ
- 判断文カバレッジ(C1:分岐網羅率)
- 単純条件カバレッジ(C2:条件網羅率)
- MC/DC (Modified Condition/Decision Coverage)

アプリケーション検証

実行時メモリエラーを自動検出

メモリ破壊、メモリーク、ポインターエラーといったC/C++特有の検出困難なエラーを単体テスト時またはアプリケーション実行時に自動的に検出します。実行時に実際に起きたエラーをレポートするため、誤検出が少なく、また、メモリアドレス、メモリーブロックサイズ、変数名などの情報とともにメモリの問題が起こった原因やソースコード中の場所を分かりやすく表示します。テストのコードカバレッジを測定するので、テストの妥当性も確認できます。

検出するメモリエラーの種類

- 不正なポインターへのアクセス
- 不正なポインターの解放
- メモリーク
- 未初期化メモリへのアクセス
- メモリ破壊 など

※C++言語については、C言語と同等のエラーのみ検出可能です。



単体テストモード

単体テストの実行中にメモリの使用状況をモニターし、メモリエラーやカバレッジをレポートします。開発の早い段階から潜在的なメモリの問題を検出できます。

アプリケーションモード

実際にアプリケーションを動作させる環境でアプリケーションを実行し、メモリエラーを検出できます。完成に近い段階で検証するので、運用後に発生する可能性の高い重大なエラーを発見、修正できます。

シミュレーター、ターゲット環境で実行可能

単体テストモードとアプリケーションモードの両方のテストを、シミュレーターや実機(ターゲット機)環境で実施できます。C++testをインストールできないターゲット環境でも実行時メモリエラー検出機能を利用できるので、実機(ターゲット機)上でのメモリエラーの発生を確認できます。また、アプリケーション実行時のカバレッジを測定することもできるので、テストの漏れを防止できます。

※詳細は、「組込みソフトウェアの開発におけるC++testの利用」をご参照ください。

機能安全規格

IEC 61508/ISO 26262 準拠:第三者試験認証機関による認証済みのC++test(Ver.9.0)でツール認証の負担なしに安全対策を効率化

C++testは、第三者試験認証機関であるTÜV SÜD社よりIEC 61508およびISO 26262に準拠したテストツールとして認証を取得しました。IEC 61508およびISO 26262は、ツールが開発プロジェクトで使用するのに妥当なものであるかを証明できない限りはと定めています。もし、この証明をユーザーが行うとすると、ツールの使用実績やツールの実行結果などのドキュメントを用意し、第三者に検証を依頼する必要があり、手間や費用がかかります。第三者試験認証機関の認証済みのC++testを使えば、ユーザー自身がツールの認証を行う必要はありません。

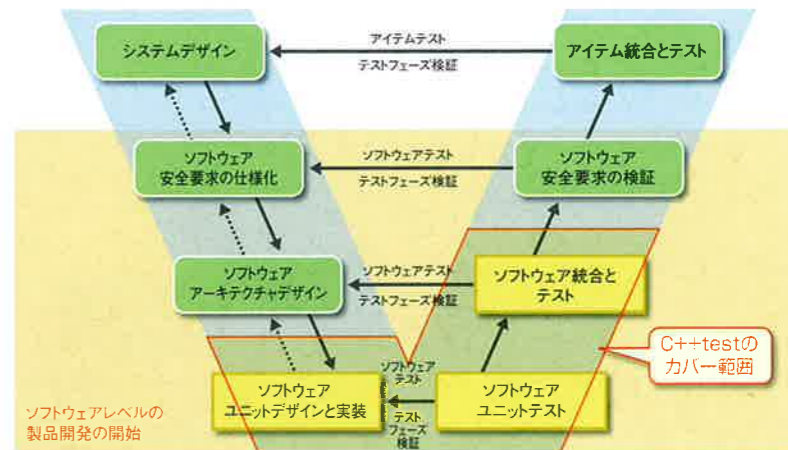


for IEC 61508
for ISO 26262

IEC 61508およびISO 26262は、システムに求められる安全水準に応じて必要な安全対策をとるよう定めています。対策には、コーディング標準に対する適合性の検証、単体テスト、カバレッジの収集などが含まれています。C++testを利用すると、これらの検証の自動化や結果レポートの出力などが可能になり、認証に準拠するための負担を軽減できます。また、単体テストやアプリケーションの検証を実機で実施することも推奨されていますが、C++testは実機でテストやカバレッジ情報の取得を行うことができます。

《ISO 26262の要求事項/技法の例》

- 静的コード解析を使用したソフトウェア実装の検証 (Table 10: 1f)
- 単体テスト実行と実行された単体テストの結果のレポート (要求事項 9.4.1)
- 境界値を用いた自動単体テスト生成 (Table 13: 1c)
- ターゲット デバイスまたはシミュレーター上の運用環境でアプリケーション検証を実行できること。(要求事項 10.4.7)



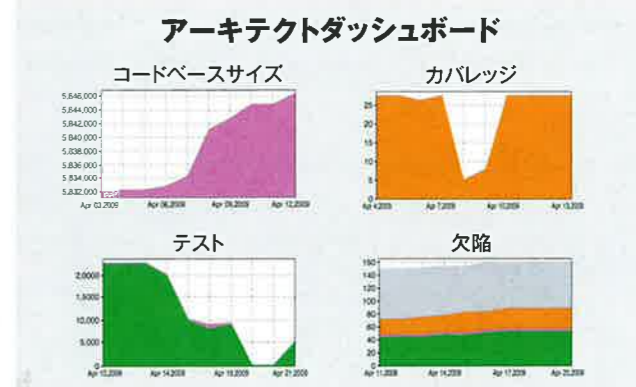
開発ライフサイクル管理ツールConcertoを利用した品質の見える化

ソフトウェア開発ライフサイクルをトータルに管理し、品質を見る化するConcertoとの連携

Concertoは、ソフトウェア開発管理のための自動化されたインフラストラクチャです。ConcertoとC++testを連携させることにより、C++testの静的解析/動的解析タスクの進捗、ソースコード作成の進捗、検証結果などを集約管理できます。さまざまな統計情報のグラフやレポートを表示し、チーム全体の品質や進捗に関する情報から、グループ・開発者レベルにブレイクダウンした詳細情報までを提供します。

※Concertoを利用するには、Concertoのライセンス(有償)が必要です。

レポートの種類	●チェックインコード量の推移	●バグの検出/修正状況
	●ビルドの結果 ●テスト結果 ●要件の実装状況	●テストカバレッジ(網羅率)分析 ●信頼性ファクター ●コードレビューの実施状況 など

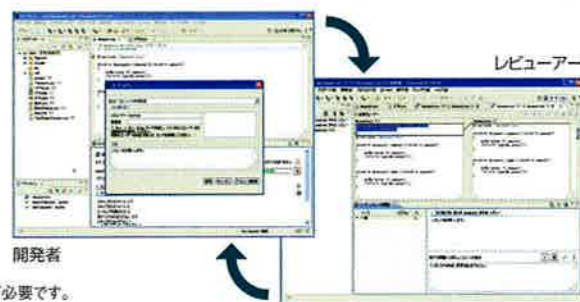


コードレビュー

円滑な目視コードレビューを実現する『コードレビュー』機能

C++testは目視によるコードレビューのワークフローを自動化し、IDEに統合された使いやすいコードレビュー環境を提供します。目視によるコードレビューは設計やロジックに関わる問題の発見に効果のある優れた手法です。しかし、タイムリーにレビューができなかったために修正に多くの工数が費やされるなど、コードレビューの利点を活かさないケースも見受けられます。C++testのコードレビュー機能を使うと、コード作成者とレビューアーがそれぞれの場所を離れることなく、自分の開発環境でいつでもレビューを実施できます。レビュー依頼やレビュー結果の通知といったワークフローが自動化されているため、開発者に余計な負担をかけません。

※「コードレビュー」機能を使用するには、Server Editionに含まれるTeam Configuration Manager が必要です。



静的解析

コーディング規約チェック

1480種類のコーディングルールでソースコードを解析

1480個のコーディングルールでC/C++ソフトウェアに潜む問題点を指摘します。コーディングルールには、ソフトウェアの品質保証や機能安全、セキュリティなどに有効なルールが含まれています。C++testを使用して、開発環境でエラーやクラッシュを引き起こす可能性のあるコードを検出、修正することにより、結合テスト以降でのテスト・デバッグ作業を軽減させることができます。

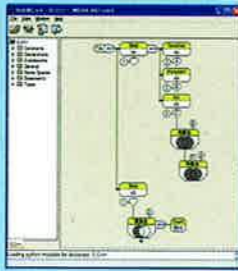
コーディングルールカテゴリ(抜粋)

- ISO 26262準拠に有効なルール
- HIS^{※1} ソースコードメトリクスに関するルール
- FDA C++ Phase 1-3(米国食品医薬局)に関するルール
- MISRA-C, MISRA-C 2004, MISRA-C++ 2008に関するルール
- OWASP TOP10, PCIDSSなど、セキュリティに関するルール^{※3}
- Joint Strike Fighterに関するルール
- SAMATE^{※2} Annex A のソースコード脆弱性(NIST Software Assurance Metrics And Tool Evaluation)アメリカ国立標準技術研究所のソフトウェア保証メトリクスツール検証
- 書式に関するルール
- メトリクスに関するルール
- バグの可能性に関するルール
- メモリおよびリソース管理に関するルール
- 移植性に関するルール
- STL ベスト プラクティスに関するルール
- IPA/SEC「コーディング作法ガイド」に関するルールなど

※1 HIS:Herstellerinitiative Softwareドイツの自動車メーカーによる団体
※2 NIST Software Assurance Metrics And Tool Evaluation:アメリカ国立標準技術研究所のソフトウェア保証メトリクスツール検証
※3 Ver.9.0 で追加されました。

コーディングルール生成ツールRuleWizard

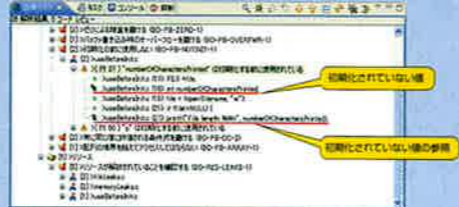
ユーザー定義コーディングルールを作成する「RuleWizard」が搭載されています。RuleWizardでは、視覚的にコーディングルールを作成したり、サンプルソースからコーディングルールを自動生成することが可能です。



静的フロー解析(バグ探偵)

静的フロー解析(バグ探偵)

C++testの静的フロー解析機能「バグ探偵」は、ソースコードを解析し、存在するすべてのパスを検出。パスの処理フローを検証し、メモリリークや、バッファオーバーフロー、未初期化メモリといった重大な問題の引き金となるコードを検出します。処理フローが複数の関数やファイルに渡った複雑なパスも検証可能なので、関数やファイル単位で実施する検証や単体テストでは発見できない場合もある問題を検出します。通常はソフトウェアを実行しなければ検出することが難しい問題を静的に検証できるため、人手によるテストに比べて網羅性の高い検証が可能となります。



- プログラムの処理フローに潜むエラーを自動検出
- 複数の関数やファイルに渡った複雑なパスも検証可能
- 問題を引き起こすコードと処理フローをレポート
- テストケースやスタブは不要。テスト資産のないソースコードでも検証可能

検出可能な問題点

- メモリリーク
- 未初期化メモリの使用
- リソースリーク
- NULLポインターの間接参照
- メモリの解放
- 配列の範囲外のアクセス
- ゼロ除算
- バッファオーバーフロー
- デッドコード
- スレッド管理
- セキュリティ関連のルール

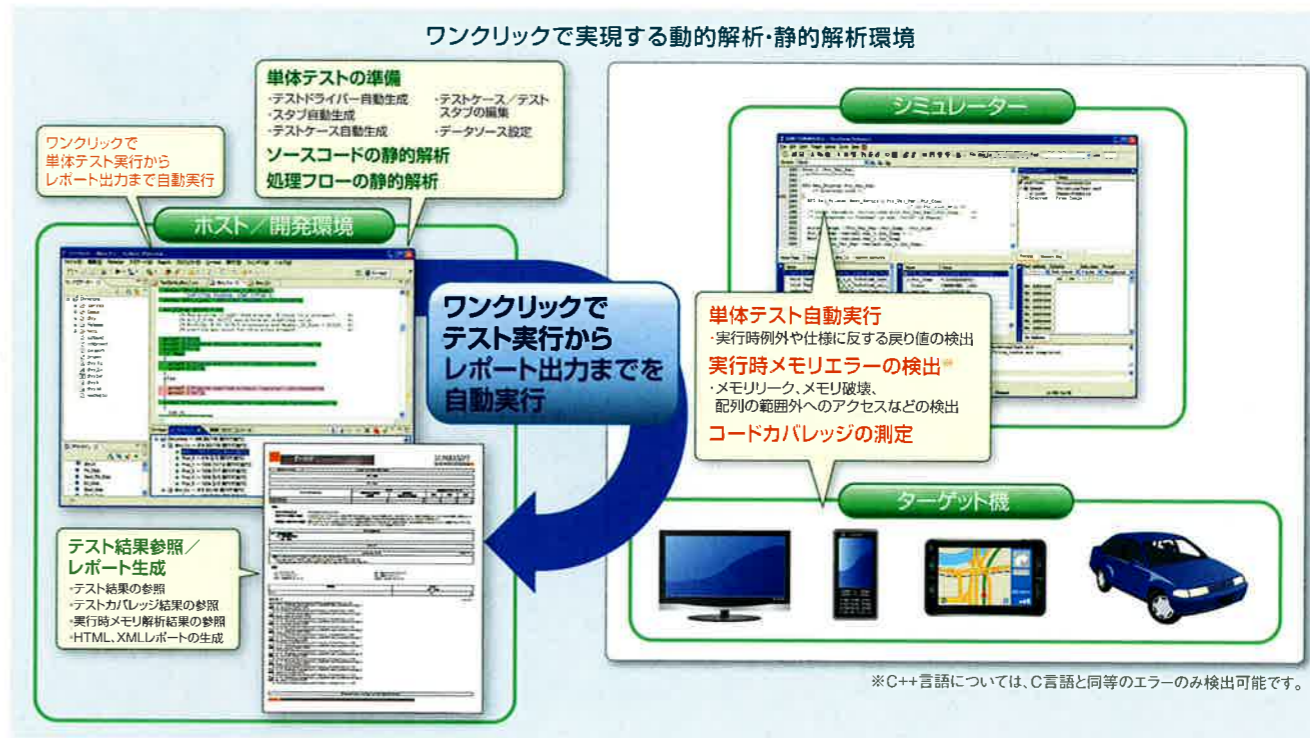
※「バグ探偵」は、Server Editionのみに含まれる機能です。

組み込みソフトウェアの開発におけるC++testの利用

実機で、シミュレーターで、単体テストと実行時メモリエラーの検出を自動実行

C++testは、さまざまなクロスコンパイラをサポートしており、C++testをインストールしたホストマシンだけでなく、実機(ターゲット機)やシミュレーター上でも単体テストと実行時メモリエラーの検出を実行できます。ターゲット環境にC++testのテスト実行用モジュールを転送し、実機(ターゲット機)上でテストを実行

した後、結果をホストマシンのC++test GUIにインポートするといった一連の作業をクリックひとつで自動実行します。また、カバレッジ情報も収集するので、テストの妥当性も確認できます。C++testを利用することにより、ターゲット環境における検証を自動化する効率的なテスト環境を構築できます。



※C++言語については、C言語と同等のエラーのみ検出可能です。

対応クロスコンパイラ/開発環境

ますます拡充、多彩なコンパイラ/開発環境をサポート

- | | | | | |
|-----------------------|--------------------------|----------------------------|----------------------|-------------------|
| • Altera New | • GreenHills MULTI | • Keil μ Vision3 | • STMicroelectronics | • Microsoft |
| • ARM RealView | • Cosmic New | • QNX | • Texas Instruments | • GCCベースのクロスコンパイラ |
| • ARM Developer Suite | • IAR Embedded Workbench | • Renesas HEW New ※ | • Wind River | |

※一部のクロスコンパイラ/開発環境は、静的解析のみのサポートです。詳しくは、稼動環境をご参照ください。

単体テスト

クロスコンパイルからテスト結果レポートの生成までをワンクリックで

テストケースやスタブの生成→クロスコンパイル→ターゲット環境へのテスト実行用モジュールの転送→ターゲット上でのテストの実行→ホストマシンへのテスト結果の転送という一連の作業をクリックひとつで自動実行します。テスト結果はホストマシンのC++testで参照でき、ソースコードの問題箇所への移動、スタックトレースやカバレッジの確認が簡単に行えます。

実行時メモリエラー検出

実機やシミュレーター上で発生したメモリ破壊などのエラーをレポート

C++testをインストールできない実機(ターゲット機)やシミュレーターでアプリケーションを実際に操作したり、単体テストを実行しながら、メモリ破壊、メモリリーク、ポインターエラーなどのデバッグが難しいエラーを検出できます。

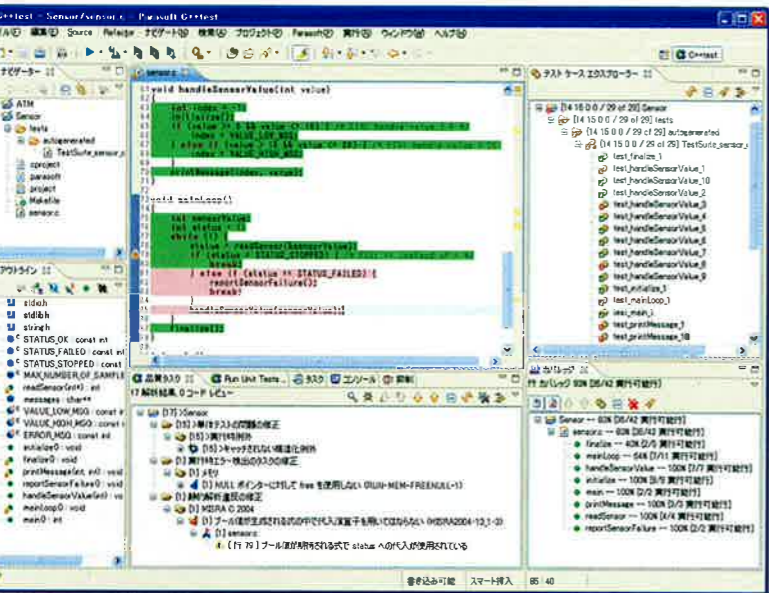
単体テストを実行しながらメモリエラーを検出

単体テストフェーズでテストケースを実行しながらメモリエラーの検出を行うと、開発の早い段階から潜在的なメモリの問題を検出できます。

結合テスト時にアプリケーションを操作しながらメモリエラーを検出

実機(ターゲット機)やシミュレーター上で実際にアプリケーションを操作しながらメモリエラーを検出できます。ユニットが結合され完成に近づいた段階でメモリエラーを検証するので、ハードウェアに組み込んだ後に発生する可能性の高い重大な問題を事前に発見できます。

※実行時メモリエラー検出は、C++言語についてはC言語と同等のエラーのみ検出可能です。※実機やシミュレーター上で単体テストや実行時メモリエラー検出を実施するには、C++test Cross Platform Editionが必要です。



レポート生成

HTML、PDF、XML形式のレポートを作成

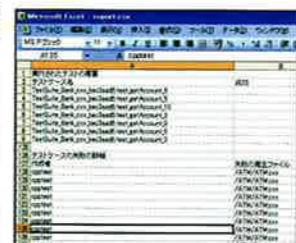
C++test GUIまたはコマンドラインから、テスト結果をHTML、PDF、またはXML形式のレポートとして出力できます。XML形式のレポートでは、XSLファイルをカスタマイズすることにより、ユーザー独自のレポートを作成できます。また、CSV形式レポートを生成するためのXSLファイルも提供されているので、テスト結果をCSV形式で出力することも可能です。



テスト結果レポート



テストカバレッジレポート



CSV形式テスト結果レポート

レポートの内容

静的解析(静的ソースコード解析とバグ探偵)

- 検出されたエラーの数
- 解析対象のファイル一覧
- ファイルごとに検出されたエラーの詳細(ルールID、ルール名、行番号)
- 有効にしたルール

動的解析(単体テストとアプリケーション検証)

- テスト実行
- 実行されたテストの概要(テストケース名、成功、失敗)
- 実行したテストケースの数
- 失敗したテストケースの詳細(テストケース名、テスト対象ファイル、行番号、スタックトレース)
- カバレッジサマリー(ファイル名、関数ごとの網羅率)
- カバレッジ詳細情報(ソースコード中の実行された行、実行されていない行を色分けで表示)

○稼動環境

Windows

- Windows 2000, Windows XP, Windows Vista, Windows Server 2003, Windows 7
- コンパイラ
 - ・ Visual C++ 6.0, .NET, .NET 2003, 2005, 2008, 2010
 - ・ GNU, MingW gcc/g++ 2.95.x, 3.2.x, 3.3.x, 3.4.x
 - ・ GNU gcc/g++ 4.0.x, 4.1.x, 4.2.x, 4.3.x, 4.4.x
 - ・ Green Hills MULTI for Windows x86 Native v4.0.x

Linux (32bit)

- Linux カーネル 2.4以上
- glibc 2.3以上
- x86 互換プロセッサ
- コンパイラ
 - ・ GNU gcc/g++ 2.95.x, 3.2.x, 3.3.x, 3.4.x, 4.0.x, 4.1.x, 4.2.x, 4.3.x, 4.4.x(x86モード)

Linux (64bit)

- Linux カーネル 2.6以上
- glibc 2.3以上
- x86 互換プロセッサ
 - ※ Linux x86_64環境でC++testを使用するには32-bit互換パッケージが必要です。
- コンパイラ
 - ・ GNU gcc/g++ 2.95.x, 3.2.x, 3.3.x, 3.4.x, 4.0.x, 4.1.x, 4.2.x, 4.3.x, 4.4.x(x86_64モード)

Solaris

- Solaris 7, 8, 9, 10 SPARC版 オペレーティングシステム
 - ※ Solaris x86版 オペレーティングシステムには対応していません。
- コンパイラ
 - ・ Sun C++ 5.3(Sun Forte C++ 6 Update 2)
 - ・ Sun C++ 5.5(Sun ONE Studio 8)
 - ・ Sun C++ 5.6(Sun ONE Studio 9)
 - ・ Sun C++ 5.7(Sun ONE Studio 10)
 - ・ Sun C++ 5.8(Sun ONE Studio 11)
 - ・ Sun C++ 5.9(Sun ONE Studio 12)
 - ・ GCC 2.95.x, 3.2.x, 3.3.x, 3.4.x, 4.0.x, 4.1.x, 4.2.x, 4.3.x, 4.4.x
 - ・ Green Hills MULTI for SPARC Solaris Native v4.0.x
 - ※ Sun CC コンパイラの -xarch=v9(64-bitモード)オプションはサポートしません。

AIX

- AIX 5.3 および PowerPC プロセッサ
- コンパイラ
 - ・ IBM XL C/C++ compiler 8.0
 - ・ GNU gcc/g++ 4.1.x
 - ※ 32-bitモードと 64-bitモードの両方がサポートされます。

システム構成

- CPU: 1.5GHz以上のx86プロセッサ (Windowsのみ)
- メモリ: 512MB RAM (1GB以上を推奨)
- インストール時に600MB以上の空き容量
 - ※ Server Editionにて、ナイトリービルド(パッチ処理)およびバグ探偵の実行を行う場合、メモリ2GB以上を推奨します。

共通項目

- クロスコンパイラ
 - ・ Altera (ホスト環境はLinuxに対応)
 - ・ Nios GCC 2.9 (静的解析のみ)
 - ・ Nios II 5.1 GCC 3.4 (静的解析のみ)
 - ・ ARM (ホスト環境はWindowsに対応)
 - ・ ARM RVCT 2.2, 3.0, 3.1, 4.0, 4.1, ARM ADS 1.2
 - ・ Cosmic (ホスト環境はWindowsに対応)
 - ・ Cosmic Software 68HC08 C Cross Compiler V4.6.X (静的解析のみ)
 - ・ eCosCentric (ホスト環境はLinuxに対応)
 - ・ GCC 3.4.x (静的解析のみ)
 - ・ Embedded Linux (ホスト環境はWindows, Linux, Solarisに対応)
 - ・ GNU gcc/g++ 2.95.x, 3.2.x, 3.3.x, 3.4.x, 4.0.x, 4.1.x, 4.2.x, 4.3.x, 4.4.x
 - ・ Green Hills (ホスト環境はWindowsとSolarisに対応)
 - ・ Green Hills optimized compilers line 4.0.x

- ・ IAR (ホスト環境はWindowsに対応)
 - ・ IAR ANSI C/C++ Compiler V5.30 for ARM (C言語のみ)
- ・ Keil (ホスト環境はWindowsに対応)
 - ・ ARM/Thumb C/C++ Compiler, RVCT3.1 for uVision
 - ・ C51 Compiler V8.18 (静的解析機能のみ)
- ・ Microsoft (ホスト環境はWindowsに対応)
 - ・ Microsoft Visual C++ for Windows Mobile 8.0, 9.0
 - ・ Microsoft Embedded Visual C++ 4.0
- ・ QNX (Windowsのみ)
 - ・ GCC 2.9.x, 3.3.x, 4.2.x
- ・ Renesas (ホスト環境はWindowsに対応)
 - ・ SuperH RISC engine C/C++ Compiler V9.03 (C++については限定サポート)
- ・ STMicroelectronics (ホスト環境はWindowsに対応)
 - ・ ST20, ST40 (静的解析機能のみ)
- ・ TASKING
 - ・ TriCore C/C++ Compiler v3.3 (ホスト環境はWindowsに対応)
 - ・ ROC198 C Compiler v6.0 (ホスト環境はWindows, Solarisに対応, 静的解析のみ)
- ・ Texas Instruments
 - ・ Windows (CCS 4.x)
 - ・ TMS320C6x C/C++ Compiler v6.1.x
 - ・ TMS320C2000 C/C++ Compiler v5.2.x
 - ・ TMS320C55x C/C++ Compiler v4.3
 - ・ TMS320C54x C/C++ Compiler v4.2 (静的解析のみ)
 - ・ MSP430 C/C++ Compiler v3.2.x (静的解析のみ)
 - ・ Windows (CCS 3.x)
 - ・ TMS320C6x C/C++ Compiler v5.1
 - ・ TMS320C6x C/C++ Compiler v6.0
 - ・ TMS320C2000 C/C++ Compiler v4.1 (静的解析のみ)
 - ・ Solaris
 - ・ TMS320C2x/C2xx/C5x Version 7.00 (静的解析のみ)
 - ・ TMS320C6x C Compiler v4.3 (静的解析のみ)
 - ・ TMS320C6x C Compiler v4.00 (静的解析のみ)
 - ・ TMS320C6x C Compiler v5.1 (静的解析のみ)
- ・ Wind River
 - ・ GCC 2.96, 3.4.x, 4.1.x (ホスト環境はWindows, Linux, Solarisに対応)
 - ・ DIAB 5.0, 5.5, 5.6 (ホスト環境はWindows, Linux, Solarisに対応)
 - ・ DIAB 5.7, 5.8 (ホスト環境はWindowsに対応)
 - ・ EGCS 2.90 (ホスト環境はWindowsに対応)
 - ・ GCC 3.3.x for VxWorks 653 (ホスト環境はWindowsに対応, 静的解析のみ)
- C++testをプラグインできる統合環境
 - ・ Eclipse 3.2(32 bit), 3.3(32 bit), 3.4(32 bit), 3.5(32 bit), 3.6(32 bit)
 - ・ Visual Studio .NET 2003, 2005, 2008, 2010
 - ・ Wind River Workbench 2.6, 3.0, 3.1, 3.2
 - ・ ARM RealView Development Suite 3.0, 3.1, 4.0, 4.1
 - ・ QNX Momentics IDE 4.5 (QNX Software Development Platform 6.4)
 - ・ Texas Instruments Code Composer Studio 4
 - ※ マネージドコードの解析には対応していません。
 - ※ Eclipse にプラグインする場合は、以下の環境が必要です。
 - ・ Eclipse IDE for C/C++ Developers 3.2(32-bit), 3.3(32-bit), 3.4(32-bit), 3.5(32-bit), 3.6(32-bit)およびEclipseがサポートするJava Runtime Environment (JRE)
- C++testにインポートできるプロジェクト
 - ・ ARM ADS 1.2
 - ・ Green Hills MULTI 4.0.x
 - ・ IAR Embedded Workbench 5.3, 5.4
 - ・ Keil RealView MDK 3.40(uVision3 3.7.2), 4.0(uVision4)
 - ・ Microsoft Embedded Visual C++ 4.0
 - ・ Microsoft Visual Studio 6
 - ・ Renesas High-performance Embedded Workshop 4
 - ・ Texas Instruments Code Composer 3.1, 3.3
 - ・ Wind River Tornado 2.0, 2.2

※ 青色のものが Ver.9.0 でサポートした環境です。

○Editionの紹介

C++test Professional Edition

動的解析

- 単体テスト
 - ・ テストケースとスタブの自動生成
 - ・ テストケースとスタブの編集
 - ・ 外部ファイルからのテストケースのインポート
 - ・ CppUnitテストケースのインポート
 - ・ テストの自動実行
 - ・ テストカバレッジ情報の測定/レポート
- アプリケーション検証^{※1}
 - ・ 実行時メモリーエラー検出^{※1}

静的解析

● 静的ソースコード解析

その他

- HTML, PDF, XML形式のレポート出力
- CVS, Subversionなどの構成管理ツールとの連携
- Concerto(別売)との接続
- コードレビュー機能^{※2}
- Team Configuration Managerとの接続

C++test Architect Edition

- C++test Professional Editionの機能
- コーディングルールの作成/編集

C++test Server Edition

- C++test Architect Editionの機能
- バグ探偵によるフロー解析
- コマンドラインインターフェイス
- Team Configuration Manager

C++test Cross Platform Professional Edition

- C++test Professional Editionの機能
- 実機(ターゲット機)/シミュレーション環境での単体テスト
- 実機(ターゲット機)/シミュレーション環境でのアプリケーション検証^{※1}
- 実機(ターゲット機)/シミュレーション環境でのテストカバレッジ情報の測定/レポート
- 実機(ターゲット機)/シミュレーション環境でのテストのレポート出力

※1 アプリケーション検証ならびに実行時メモリーエラー検出機能は、オプション(無償)機能です。
 ※2 コードレビュー機能を使用するには、最低でもチーム内にServer Editionが1ライセンス必要です。

【開発元】

PARASOFT®

We make software work.™

C++testの情報は

<http://www.techmatrix.co.jp/quality/ctest/>

● 掲載されているあらゆる製品名は、各社の商標あるいは登録商標です。

【総販売代理店】

TechMatrix

テクマトリックス株式会社

システムエンジニアリング事業部

ソフトウェアエンジニアリング営業部

本社: 〒108-8588 東京都港区高輪 4-10-8 京急第7ビル

TEL.03(5792)8606 FAX.03(5792)8706

大阪営業所: 〒541-0054 大阪市中央区南本町 2-6-12

サンマリオンNBFタワービル

TEL.06(6243)3801 FAX.06(6243)3803

[URL] <http://www.techmatrix.co.jp>[E-mail] parasoft-info@techmatrix.co.jp

R100

日本製率100%再生紙を
 採用しています。



このカタログの印刷には、環境に配慮した
 植物性インキを使用しています。