ランタイムエラーが存在しないコードを証明する Polyspace

高品質ソフトウェアの開発

高い信頼性を実現するためには

組み込みシステムは非常に高い信頼性が要求されます。しかし信頼性を劣化させる要因であるランタイムエラーを除くためには、多大なテストやレビューが必要になります。

発見の困難なランタイムエラー

ランタイムエラーとはコンパイル時には分からずに、プログラム実行時に発生するソフトウェアの不具合です。例えば、演算した結果、型の値の範囲を超えてオーバーフローが発生するというエラーです。このような不具合は稀にしか発生せず再現性に乏しいなど、テストでの発見が難しい場合が多く、出荷後の製品で不具合が発生した場合、製品回収などにつながるおそれがあります。

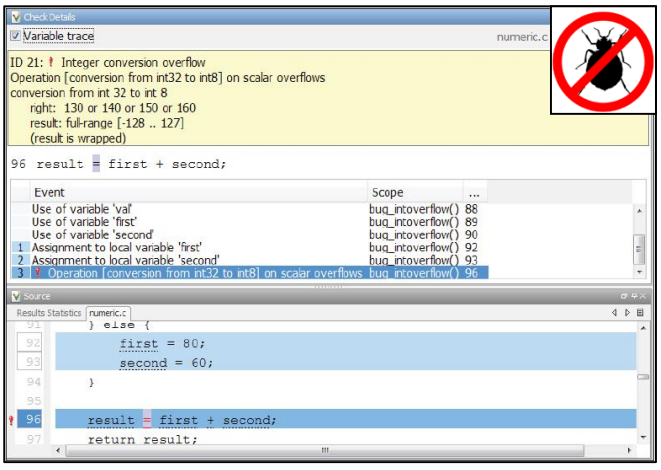
ランタイムエラー検証技術により高信頼性を実現

Polyspace Bug Finder™

C および C++ 組み込みソフトウェア内のランタイム エラー、データフローの問題、およびその他の欠陥を特定します。Polyspace Bug Finder による静的解析では、ソフトウェア制御、データフロー、手続き間の動作を解析します。これにより、**開発プロセスの初期段階においてバグの選別と修正が可能になります**。その上、Polyspace Bug Finder を使用してコーディングルール(MISRA、JSF、命名規則) 違反の検出を行うことで、コーディングルールへの準拠をチェックします。

Polyspace Code Prover™

静的解析と形式手法に基づいた抽象解釈を使用しており、手書きのコード、生成されたコード、またはこの 2 つの混合コードに使用できます。**ランタイム エラーが発生する可能性のあるコードと、ランタイム エラーがないことが証明されたコードを特定します**。また、Polyspace Code Prover は変数および関数の戻り値の範囲情報を表示し、変数が指定された範囲を超えるケースを特定することもできます。



Polyspace Bug Finderによるオーバーフローの指摘

レビュー、テスト手法の限界を克服

静的解析の限界を克服

コンパイラや静的ツールではソースコードの構文から文法違反の箇所やコーディングルールに沿っていない箇所を指摘することが可能です。しかし動的な振る舞いが関係するランタイムエラーは見つかりません。Polyspace® はコンパイラや従来型の静的テストツールでは検出されなかったランタイムエラーを静的解析のみで検出します。

レビューの限界を克服

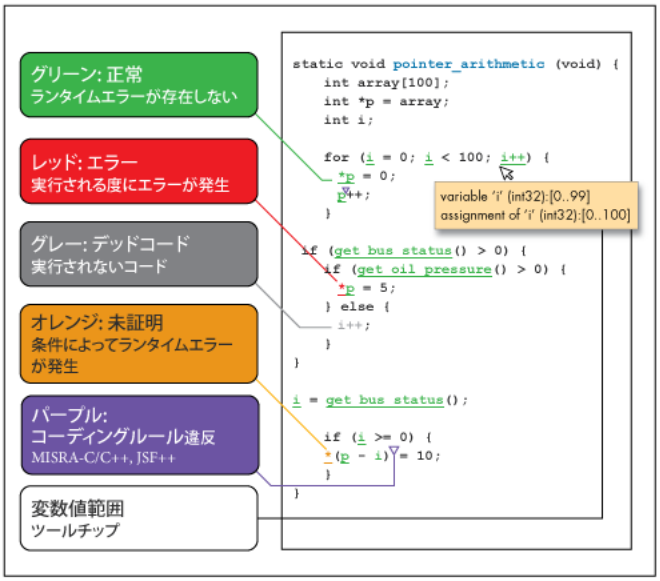
ソースコード規模が膨大、複雑になるとレビューは難しくなり、作業工数の増大、見落としが発生します。Polyspace Code Proverのコード証明によりレビューの作業を大幅に削減することが可能です。

- 緑で表示される箇所では該当する検査対象エラーは発生しません。ランタイムエラーの観点でのレビューを行う必要はありません。
- 開発者は結果をレビューすることにより、ランタイムエラーが発生する可能性のある脆弱性の把握ができます。

動的テストの限界を克服

動的テストでもランタイムエラーは見つかりますが、特定の条件でしか起きないようなエラーなど、テストケース不足でランタイムエラーが発生せず、発見できない可能性があります。その上、ランタイムエラーが発生しても現象に気付かずに見逃す可能性もあります。Polyspaceを使用することで、下記のような検証作業の向上が可能となります。

- 抽象解釈を応用した網羅的な静的解析により、テストケース漏れやバグ発生に気づかずにランタイムエラーを見逃す危険性を大幅に削減します。



Polyspace Code Prover によるランタイムエラー証明

Polyspace静的解析のメリット

コード証明による品質測定・管理・向上

すべてのランタイム条件を完全かつ総合的に検証し、ランタイムエラーがないことを証明

素早いバグ検出

コード作成・更新・編集の直後にバグやセキュリティー脆弱性の検出・デバッグ・修正が可能。

ランタイムエラーの高度な解析

形式手法により、動的テストの前に潜在的エラーを検出

品質保証プロセスの短縮

開発プロセスの上流でランタイムエラーを検出・修正し、ソフトウェア品質の管理・向上が可能

リリース前により多くのバグを検出

静的解析により、バグ検出プロセスを効率化

ロバスト性・信頼性を確保

動的テストやコードレビューでは不可能な、コード内の全ての変数値や演算結果を検証

静的解析ソリューション

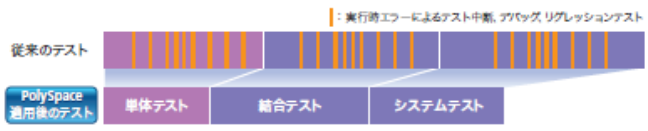
コードメトリクス測定・コーディングルールチェック・欠陥検出・ランタイムエラー証明を可能とし、豊富で便利なデバッグ機能を用いた静的解析ソリューション

テスト工数を削減、テスト期間を短縮

テスト工数削減効果

- テストを行うには、テストケースを作成し、実行環境でテストをする必要があります。アプリケーションのサイズが増加するにつれ、テスト工数が増大してしまいます。テスト時にランタイムエラーが発生すると、デバッグ、リグレッションテストにかかる時間が増加し製品開発の大きな妨げとなります。
- 開発の後工程でバグが発見されるほど修正にかかるコストは大きくなります。開発の早い段階で不具合を見つけることがコスト削減につながります。
- エラー検出の対象となるソフトウェアの複雑性が増すと、テストケースの数も増加します。

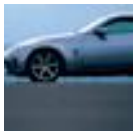
早い段階でランタイムエラーを取り除くことにより、レビュー、テスト工数の削減が可能となります。テストケース作成不要、実行不要で不具合の早期検出を可能とするPolyspaceを活用して開発期間・コストの短縮を実現できます。



ユーザー事例

フォード

フォードは自社の静的解析ツールをPolyspace 製品に置き換え、それを幅広い開発チームに展開しました。



日産自動車

「Polyspace 製品によって、高レベルなソフトウェアの信頼性を確保することができます。これは、業界内の他のツールにはできないことです。」

デルファイ ディーゼル システムズ

「Polyspace製品をなるべく早くソースコードに適用することで、バグを早期に検出しコストを削減することができます。」

EADS

「Polyspaceのようなソリューションは他にありません。コードを実行せずにランタイムエラーを検出でき、その網羅的なコード解析は大きな強みです。」

ソーラー・インパルス

Polyspace製品を使用して早期に、かつ素早く問題を発見し取り除くことで、ソーラー・インパルスは12～24人月分の開発時間を短縮し、DO-178準拠という目標を達成することができました。

リソース

製品詳細、使用例、システム要件
mathworks.com/polyspace

営業へのお問い合わせ
mathworks.com/contactsales

サポートへのお問い合わせ
mathworks.com/support

Polyspace担当エンジニア
能戸 フレッド
Fred.Noto@mathworks.co.jp