PolySpace for C

ソフトウェア品質 信頼性向上への新たなアプローチ

高信頼性、高機能化かつ省コスト化の相反する要求への挑戦

組み込みシステムは非常に高度な信頼性が要求され、更にプログラムのサイズと複雑さの増大はテストのコストの増加を招きます。現行のテストツールや手法ではこれらの要求に対応できなくなってきています。
事実、ランタイムエラーを検出するためには従来の手法は適切ではありません。例えば、開発プロセスでのメトリック測定やコーディングルールの適用などはソフトウェアの品質を上げる良い方法ではありますが、不具合そのものの検出には至りません。動的テストは要求仕様、設計の検証には欠くことのできない手法ですが、作成されたテストケースによるテスト実行箇所のみの不具合しか検出されません。また従来の手法では検出された不具合をデバッグする為に多大な時間を費やすこととなります。

現在のソフトウェア開発においてはソフトウェア品質向上と開発コスト削減を兼ね備えた新世代のテストソリューションが求められています。

新たなアプローチ

PolySpace は開発の早期にテストを実行せずにランタイムエラーや共有データへの同時アクセスを自動的に検出します。
ソースコードからソフトウェアの動的な特性を静的に解析します。

- テストケースを作成する必要はありません
- コードのインストゥルメントは必要ありません
- プログラムを実行する必要はありません

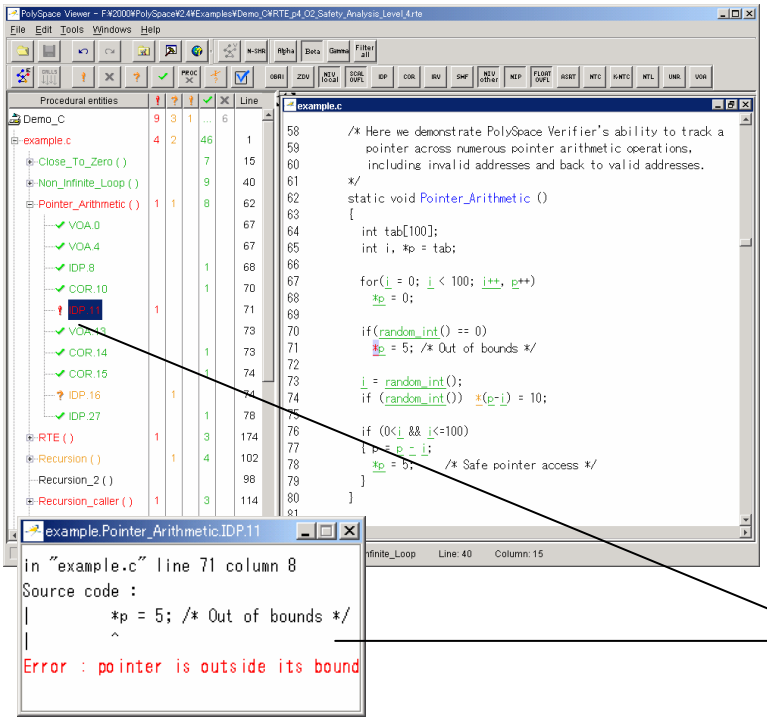
動的テストに比べ PolySpace は自動的にソースコード中のランタイムエラーの箇所を明確に表示します。従って、短時間で不具合の検出が可能となります。
PolySpace を使用すればテストおよびデバッグに要する時間を大幅に短縮し、その結果、ソフトウェア開発コストの削減に大きな効果をもたらします。

ランタイムエラーの検出

ランタイムエラーとはプロセッサの停止、データの破壊、セキュリティホールなどの結果を引き起こす不具合です。これらは、通常実行時に表面化するソフトウェアの潜在的な欠陥であり、テストの予算オーバー、リリースの遅れ、操業中のサービスの中断、製品回収等、ビジネスに大きな影響を与えます。

PolySpace により検出されるランタイムエラー:

- ヌルポインタやポインタの境界外への不正な参照
- 配列の境界外へのアクセス
- 零除算や負数の平方根演算等の不正な数学演算
- 整数や浮動小数点数へのオーバーフロー、アンダーフロー
- long から short、float から int などの危険な型変換によるエラー
- 未初期化変数へのアクセス
- スレッド間で共有される変数への同時アクセス
- 終了しない関数や無限ループ
- デッドコード 他



レッド	エラーが確実に発生します。
グリーン	エラーは発生しません。
オレンジ	エラーの発生する可能性があります。
グレー	デッドコード

静的検証

動的特性のソースコードのみによる静的検証は、与えられたソフトウェアの全ての可能性のある実行を表す理論 (抽象解釈 (Abstract Interpretation)) として既に確立されています。この理論はスレッド間、関数間そして制御構造内の解析をプログラムの各ポイントに適用し、各々の変数の取り得る値の領域を計算するという考えに基づいています。この理論を応用してランタイムエラーを検出する自動的なアプローチにより、PolySpace はソフトウェアのテストや出荷に影響を与えるランタイムエラーを静的に予見できる市場で唯一のツールです。
例えば以下のソースコードでは

```
int tab [100];
int *p = tab;
int i;
for (i = 0; i < 100; i++, p++)
   *p = 0;
*p = 5;  /* ランタイムエラー発生！ */
```

`*p = 5;` の箇所でメモリ破壊が発生しますが、PolySpace による自動検出が可能です。その一方、その他の全てのテスト手法では同じエラーを検出する為に人手 (コードレビュー、テストの実行、コードインストゥルメンテーション等) が必要とされます。

利点

PolySpace は組み込みソフトウェアの開発者と品質保証エンジニアに品質と生産性の向上をもたらします。

PolySpace Developer エディション

開発者は各テスト工程 (単体/統合/システムテスト) の前に不具合を早期発見する恩恵を得られるので、テストとデバッグ時間の短縮においては非常に重要な要素になります。PolySpace は最も迅速かつ自動的にランタイムエラーを検出します。実際、ソースファイルをクリックするだけで結果が得られます。テストケースを作成する必要は無く、インストゥルメンテーションをする必要も無く、プログラムを実行させることもありません。従って PolySpace による導入効果が直ちに現れます。PolySpace により開発の早期工程で検出された不具合が、もしテストの後期工程で発見されたならば、かなりの デバッグ時間がかかるはずです。さらに PolySpace により解析済みの関数は堅牢性が高いので、後のテスト工程は容易になります。

PolySapce Auditor エディション

品質保証エンジニアは、サプライヤとエンドユーザーの関係において最も重要な要素となる、ソフトウェアの品質に関する信頼できる評価を得られます。PolySpace はコード開発フリーズ後のシステムテストや納品後に確認されるような不具合の数を劇的に減らすことにより、ソフトウェアの品質を自動的に向上させます。PolySpace は内部開発の品質管理と外注開発の納入検証の双方に適用可能です。

PolySpace は不具合をより多く、迅速に検出する新しい手法です。