実行トレース解析プラットフォームPolyspector™

ソフトウェア実行トレース情報の解析により開発状況を自在に可視化

ソフトウェアの詳細な実行履歴に関する情報は生成される量が膨大なため，ソフトウェアの品質を解析する材料として十分には活用されていませんでした。しかし，近年の技術革新によりストレージや計算資源が安価になりつつあり，詳細な実行トレース情報を低コストで蓄積して解析するための環境が整いつつあります。

そこで，ソフトウェア実行トレース情報を蓄積して解析し，可視化するための統合プラットフォームPolyspector™を開発しました。これにより，ソフトウェアの問題点や改善箇所を，ソフトウェア実行トレース情報から開発者に効率よくフィードバックすることが可能になります。

ソフトウェア開発の課題

近年，製品の付加価値を決める要素としてソフトウェアの重要性が高まってきており，この10年では自動車や携帯電話などの組込み機器に搭載されるソフトウェアの規模は5～10倍になったと言われています。これに伴い，ソースコードの品質を解析するツールや，毎日のテスト作業を自動化するツールなど，様々なツールが登場しソフトウェア開発の生産性向上に寄与してきました。

しかし，スキルや経験に大きく依存する作業，例えば複雑に絡み合ったバグの原因を見つけ出す，あるいは様々な選択肢の中からもっとも費用対効果の高い修正案を選択する，といった作業については自動化が難しく，今も人手に頼っているのが現状です。

実行トレース情報の活用

このような作業を人はどのようにこなしているのでしょうか。スキルの高い開発者は，特定の場面でソフトウェアがどう動作するのかを暗黙的に理解しており，そのような知識に基づいて効率的にバグ原因を絞り込んだり，修正案が引き起こす変化を予想したりしていると考えられます。

そこで，実行トレース情報と呼ばれる，ソフトウェア実行中に発生した分岐やメモリアクセスなどを時系列に記録した実行履歴に注目しました。実行トレース情報はソフトウェアの実行時間に比例してデータ量が増えるとても大きなデータですが，内部状態遷移や，処理時間，メモリアクセスパターンなど，ソフトウェアの品質を解析するうえで有益な情報が数多く含まれています。実行トレース情報を合理的かつ効率的に活用するための仕組みを開発し，主にバグの解決や性能チューニングにおいて，従来は人手に頼らざるをえなかった作業の自動化を実現したいと考えています。

実行トレース情報を合理的かつ効率的に活用するために，統合プラットフォームPolyspector™（**図1**）を開発しました。

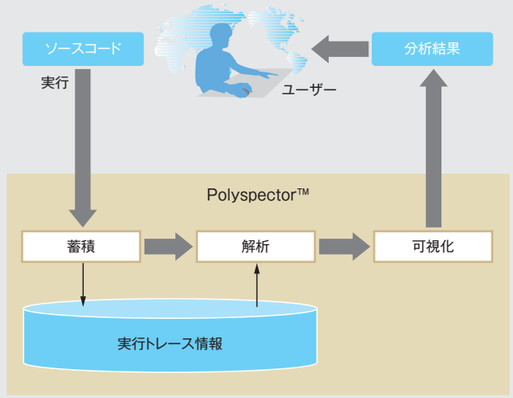

図1．Polyspector™のモジュール構成 ― ソフトウェアの実行をモニタリングして実行トレース情報を蓄積するモジュール，蓄積した情報を解析するモジュール，及び解析結果を可視化するモジュールで構成されます。

Polyspector™の特徴

実行トレース情報を活用するには，ソフトウェアの実行をモニタリングして実行トレース情報として蓄積する機能，蓄積した情報を解析する機能，及び解析結果を開発者にわかりやすく可視化する機能の三つの要素が必要です。しかし既存ツールでは，各要素をツールごとに個別に開発しており，せっかくの有益なトレース情報を断片的にしか利用できていませんでした。

そこで，様々な目的に対して幅広く対応できるよう各要素を設計し，プラットフォーム化しました。Polyspector™は，要素ごとに以下の特徴を備えます。

(1) 蓄積する　実行トレース情報を分散データベース形式で構造化して管理します。大量の情報の中から任意の情報にすばやくアクセスできるだけでなく，大規模かつ安価な分散ストレージによってデータ保持コストも削減できます。

(2) 解析する　複数の実行トレース情報間の差分に着目します。メモリアクセスに関する差分や，処理時間に関する差分など，様々な視点での差分抽出を容易に実現可能としました。具体例については後述します。

(3) 可視化する　様々な解析結果を統一的に扱うためのGUI（グラフィカルユーザーインタフェース）を提供します。現在はソフトウェアの品質を，全体を見渡して解析するためのモジュールを提供しており，ソフトウェアの階層構造マップ上に任意の解析結果を重ね合わせて表示することができます。

これらの特徴により，蓄積された実行トレース情報から，様々な解析を低コストで実現できるようになりました。

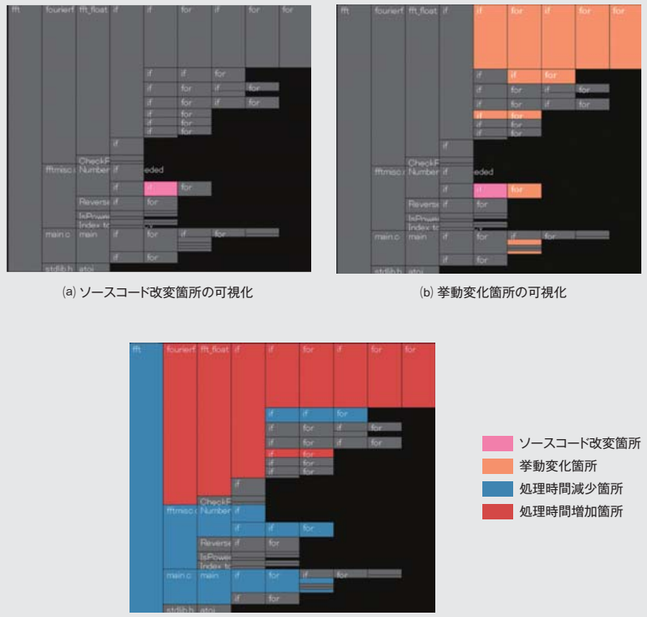

図2．バージョン間差分解析ツールの画面スナップショット ― ソフトウェアの階層構造マップ上に，ソースコードの改変に伴う品質の変化を，ユーザーの目的に応じて可視化することができます。

応用例

差分解析技術の具体例として，バージョン間差分解析ツールを**図2**に示します。このツールでは，ソフトウェアの階層構造マップを示したうえで，ソースコードの改変に伴うソフトウェア品質の変化を，ユーザーの目的に応じて可視化します。

マップは図2(a)のように，左から右に向かってアプリケーション，ソースファイル，関数，関数内処理（if，forなど）…と階層化した構造をとっています。図2(a)～(c)は，異なる3種類の差分解析結果を示しています。

(a)では，ソースコードの改変箇所（ピンク色）を表示しています。

(b)では，条件分岐の挙動が変化した箇所（オレンジ色）を(a)の結果と重ねて表示することで，改変された影響で挙動が変化している箇所を可視化しています。

(c)では，処理時間が減少した箇所を青色，増加した箇所を赤色で表示しています。図の左端に表示される階層構造の最上位にあるソフトウェア全体は処理時間が減少しているものの，赤色表示されたモジュールで増加していることがわかります。

このようにこのツールによって，三つの異なる視点から，ソースコードの改変がソフトウェアに与える影響を可視化できるのがわかります。

今後の展望

実行トレース情報から，ソフトウェア品質を解析するうえで有益な情報を合理的かつ効率的に抽出するためのプラットフォームを実現しました。応用例として挙げたバージョン間差分解析ツール以外にも，バグ原因の自動推定ツールや，性能向上プラン推薦ツールなど，このプラットフォームを活用した様々な応用ツールの開発を進めています。

松崎　秀則
研究開発センター
コンピュータアーキテクチャ・セキュリティ
ラボラトリー主任研究員