6A-4

# 経験に基づく推測を排除したデバッグ手法の提案

## ～デバッグ用プログラムトレーサの開発～

氏名： 射手矢 良太† 赤堀 文隆† 山崎 雄大‡ 榊原 正天‡ 古宮 誠一‡

所属： 芝浦工業大学† 芝浦工業大学大学院‡

### 1．はじめに

ソフトウェア開発には，バグが必ずといっていいほど存在する．そのバグを取り除くためには，バグが存在するかどうかを確かめるテストと，実際にバグの原因を特定して修正するデバッグという二つの作業がある．これらの作業はソフトウェアの品質を保証するために重要な作業であり，必ず行わなければならない作業である．[1]

本研究はデバッグに焦点を置いたものである．

### 2．通常のデバッグ手法とその問題

図１に通常のデバッグの手順を示す．

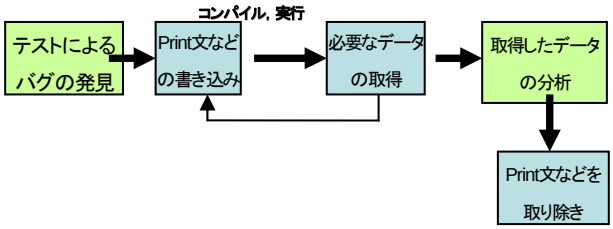


図１：トレーシングによるデバッグの手順

この手法は，例えばプログラムの動作を確認するためにPrint 文を埋め込むといった方法である．

この手法には以下の４つの問題点が存在する．

１． トレースコード書き込みの際，トレースコードをどこに書き込むか，何をトレースデータとして取得するかなどを決める際に経験的な推測を必要とする．

２． トレースデータ取得の際，バグを再現させなければならない．例えば，24 時間周期で発生するバグを任意に再現させることは，バグの発生条件が分からない状況では不可能に近い．

３． 求めるデータをトレースできるまで，トレースコード書き込みとトレースデータ取得の作業を繰り返し行わなければならない．

４． Print 文などを書き込むことは元のプログラムを変更することであり，元に戻す作業が必要になる．

### 3．問題解決のために提案するトレーサ

上記の問題を解決するために，我々はプログラムトレーサというものを提案する．トレーサとは，プログラムの実行軌跡を追うツールのことである．トレーサを次のような方針で開発する．

１．考えられる全ての情報をトレースする処理を，被デバッグプログラムのコードを変更することなく自動的に埋め込む．こうすることで，トレースコード書き込みの際の推測を排除し，加えて繰り返しの問題も解決する．更にプログラムに変更を加えないので，挿入した Print 文などを取り除く作業も不要となる．

２．テストと同時にトレーシングを行う．こうすることにより，テスト失敗時のトレースデータがすでに存在するため，バグを再現させる必要がなくなる．

このような手法を用いることにより，デバッグの手順は以下の図２のような流れとなる．

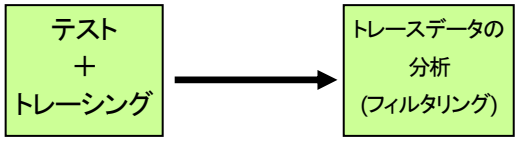


図２：我々の提案するプログラムトレーサを使用したデバッグの手順

ただし，考えられる全ての情報をトレースしてしまうと副作用として，情報の過剰取得によるトレースデータの可読性の低下が発生してしまうので，トレースデータの分析時にフィルタ処理を行う．

尚，今回作成するトレーサは Java で書かれたプログラムのみを対象とする．

### 4．Tracer と Filter

今回作成するツールのうち，実際にトレーシングを行う部分を Tracer，フィルタ処理を行う部分を Filter と名付けて開発を行った．

Tracer は，「考えられる全ての情報」という概念を表

---

Proposal of a Method for Debuging an Object-Oriented Program without Reasoning based on Experiences

†Ryota Iteya, Fumitaka Akahori

‡Yudai Yamazaki, Masao Sakakibara, Seiichi Komiya

†Shibaura Institute of Technology

‡Graduate School of Shibaura Institute of Technology

現するために取得できる情報を逐一拡張できるような方式をとり，トレースデータのバイナリを出力する．

Filter は，ユーザに設定ファイルでフィルタリングの際の設定を行ってもらうという手法をとり，Tracer の出力を入力とし，フィルタ処理を行ったトレースデータの txt ファイルを出力する．フィルタリングの処理の様子を以下の図３に示す．図の左側が Tracer が出力した全てのトレースデータで，右側がフィルタリング処理を行った結果の出力である．図の例では，Point クラス内の move メソッドのシグネチャ及び引数のみ表示するようなフィルタリングを行った例である．

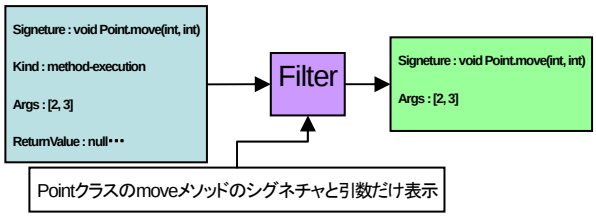


図３：Filter によるフィルタリング処理

５．アスペクト指向プログラミング

現在，本研究では被デバッグプログラムのコードを変更することなくトレースコードを埋め込むという処理を実現するための手法の一つとして，アスペクト指向プログラミングを使用している．アスペクト指向プログラミングとは，例えばロギング処理のようなオブジェクト指向ではうまくモジュール化できない処理をアスペクトという新たなモジュールを用意してそこに記述しようという考え方である．[2]

これを用いることで，被デバッグプログラムのコードを変更することなくトレースコードを挿入することが出来る．

６．研究状況と実験

現在，拡張性は考慮していないプロトタイプの開発を行っている．

Tracer は，トレースコードを自動的に埋め込むという方針を実現するための技術の一つとして，Java 言語を拡張したアスペクト指向言語である AspectJ を用いて開発した．

Filter は，現在では設定ファイルはテキスト形式のファイルとなっている．

現状として，研究室内に存在した GA のプログラムに対して我々のプログラムトレーサを適用する実験を行ったが，Tracer の出力したトレースデータが４００MB を超える大容量となりパフォーマンスが大変悪く，かつ Filter が読み込めないという課題が浮き彫りになった．

７．今後の課題

Tracer に関しては，現状では一定の規模以上のプログラムに対して適用すると出力されるバイナリの容量が非常に膨大になり，Filter での読み込み処理に失敗してしまうので，出力ファイルを分割することで解決を試みる．また，パフォーマンスが大変悪いので，これに対しても解決策を模索していく．

更に現時点で拡張性が全く考慮されていないので，パッチやプラグインなどの機能を実装することによって拡張性を考慮していく．

Filter に関しては，設定ファイルがテキスト形式になっているので，xml 形式などにすることによってより柔軟なフィルタ処理を行えるようにしていく．

８．関連研究

東京工業大学の薄井義行氏らが，アスペクト指向を用いて被デバッグプログラム中の任意の場所にデバッグコードを挿入する手法を提案している．[3]

この研究では任意の場所にデバッグコードを挿入するという作業を自動化しデバッグ作業を援助しているのに対し，我々の研究では任意の場所ではなく，考えられる全ての場所にデバッグコードを挿入し，トレーシングを行った後にフィルタリングしているため，デバッグコード挿入時の経験に基づく推測を必要としない．更に我々の手法ではテストと同時にトレーシングを行うため，バグの再現性を考える必要もなくなる．

武蔵工業大学の大澤直樹氏らが，アスペクト指向技術を用いてあらかじめ用意されたデバッグコード挿入箇所のテンプレートから挿入位置を選択できるデバッグツールを提案している．[4]

この研究ではデバッグコードを挿入できる位置がテンプレートによって用意されているためにデバッグコード挿入可能な箇所が限られてしまうのに対し，我々の研究では考えられる情報は全てトレーシングするため，よりデバッグに必要な情報を取得することが出来る．

９．参考文献

[1] Glenford J. Myers 著，長尾真 監訳，松尾正信 訳 “ソフトウェアテストの技法” 近代科学者 1980

[2] 千葉滋 著 “アスペクト指向入門” 技術評論者 2005

[3] 薄井義行，千葉滋 著 “アスペクト指向を用いてデバッグコードを挿入できるソフトウェア開発” 東京工業大学 2003

[4] 大澤直樹，横山孝典 著 “アスペクト指向技術を活用した C++向けデバッグツール” 武蔵工業大学 2004