4L-8

# 協調動作するオブジェクト群に着目したオブジェクト指向プログラムの実行履歴分割手法の改良

渡邊 結† 谷口 考治† 石尾 隆†‡ 井上 克郎†

†大阪大学大学院情報科学研究科 ‡日本学術振興会特別研究員 (PD)

## 1 前書き

オブジェクト指向言語で記述されたプログラムでは，実行時に動的に決定される要素が多く，その解析には実行履歴のような動的情報が有用である．しかし，実行履歴はそのサイズが巨大化しがちであり，一括して取り扱うのが困難である場合が多い．そこで，実行履歴を複数のブロックに分割して解析することが考えられるが，この際の各ブロックはプログラムの各機能に対応していることが望ましい．そのため，実行履歴を実現されている機能の単位で自動的に分割する手法が求められている．

本研究では，実行履歴上に現れるオブジェクト群の変化に着目して実現されている機能の切り替わりを検知する手法 [1] を拡張・調整し，分割点の決定と分割粒度の制御を行う．

## 2 実行履歴分割手法

実行履歴を機能単位で分割するためには，まず実行履歴上での機能の切り替わりを検知する必要がある．更にそこから，ブロックの分割点を正確に一点に決定しなければならない．また，プログラムで実現されるある機能は更に細かい複数の機能から成り立っていると考えられるため，分割の粒度は検出したい機能の粒度に応じて制御できることが望ましい．

### 2.1 機能の切り替わりの検知

実行履歴上での機能の切り替わりの検知には，既に提案されている「協調動作するオブジェクト群に着目した実行履歴分割手法 [1]」を用いる．

実行履歴上において，プログラム中の異なる機能を実現している場合，そこに登場するオブジェクト群も機能毎に異なると考えられる．そこで，実行履歴上に登場するオブジェクト群の時間変化を捉えることで，実行履歴上での機能の切り替わりを検出する手法が提案された．

具体的には，まず実行履歴上に登場するオブジェクトを時間順にサイズ $size$ のキャッシュ $C$ へ重複なしで記憶してゆく．このとき，キャッシュ溢れが生じるか否か（1,0）を記録し，更に過去 $n$ 回分のイベントでキャッシュが更新された割合（以下，更新頻度と記述する）を計算する．そして，この更新頻度がある閾値 $threshold$ 以上である時，実行履歴上で異なる機能へ移行したとみなす．

図 1 はある実行履歴における更新頻度の時間変化を表している．なお，点線は実行履歴上で機能の切り替わる点を表している．このように，この提案手法において，実行履歴上において更新頻度のピークとなる箇所は実際に機能の切り替わる点に対応しているという事が，適用実験によって確認されている．

### 2.2 分割点の決定

実行履歴を機能単位のブロックに分割するためには，履歴上で実現される機能が切り替わる箇所において，分割点を一点に定めなければ成らない．またこのとき，分割点は履歴上で実現される機能が切り替わる点を正確に反映している必要がある．

2.1 で述べた手法において，更新頻度のピークは，実行履歴上で機能が切り替わる正確な時点より遅れて生じる（図 1 拡大部参照）．これは，動作オブジェクト群の変化として過去のイベントでのキャッシュ溢れを参照した計算を行っているためである．また，更新頻度が閾値 $threshold$ より大きくなる箇所を更新頻度のピークとして検出しているため，ある機能が切り替わる点に対応する更新頻度のピークは複数イベントにまたがって検出される場合がある．これらの理由から，更新頻度のピークとなる点をそのまま分割点とすることは適切ではない．

機能の切り替わる点は，ある機能を実現する処理が全て終了し，また別の機能を実現する処理が開始される境界である．このため，履歴上のコールスタックの深さの変化をみると，機能の切り替わる点で極小値をとる予想できる．そこで，更新頻度のピークを検出し

---

**Improved division method of Object Oriented Trace Based on Collaboration of Objects**

Yui WATANABE† Kouji TANIGUCHI† Takashi ISHIO†‡ Katsuro INOUE†

†Graduate School of Information Science and Technology, Osaka University ‡Research Fellow of the Japan Society for the Promotion of Science

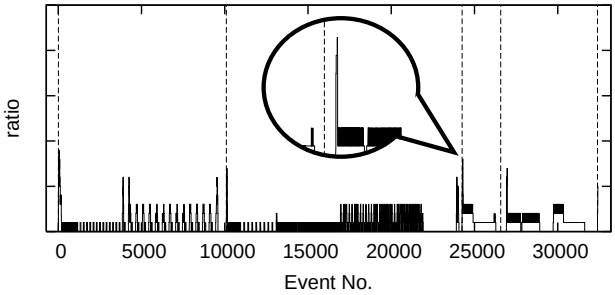


図 1: 更新頻度の時間変化例

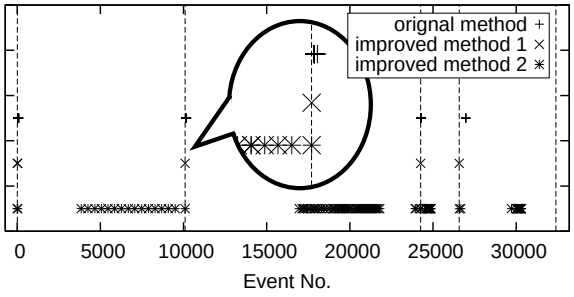


図 2: 更新頻度から検出する分割点と本手法が検出する分割点の例

た際，その時点から遡って直前にある「コールスタックの深さが極小となる点」を分割点として決定する．これにより，ある機能の切り替わる点に対応する複数の更新頻度のピークに対して，分割点はそれらの直前にある一点に定める事ができる．

### 2.3 分割粒度の制御

提案手法はキャッシュサイズ（$1 \leqq size \leqq$ 登場オブジェクト総数），ウィンドウサイズ（$1 \leq n \leq$ イベント総数），及び閾値（$0 \leq threshold \leq 1$）の 3 つのパラメータを持ち，それぞれのパラメータの値が大きいほど，検出される分割の粒度も大きくなると考えられる．そこで，各パラメータ値の変化を組み合わせる事により，機能粒度に沿った分割粒度の制御を行うための調整を行う．

## 3 適用実験

Java プログラムの実行履歴（オブジェクト情報を含むメソッド呼び出し列）に対する適用実験を行い，検出結果としての分割点の正当性の検証および各パラメータの分割粒度に対する影響の調査を行った．

### 3.1 検出結果の正当性

用いた実行履歴について，プログラムのユーザと開発者の手で機能が切り替わる点をそれぞれ挙げてもらい，これを検出の目標とする正確な分割とした（図に点線で示す）．

図 2 は，更新頻度の値のみで検出した分割点（original method）と，本手法で検出した分割点（improved method 1）それぞれについて，正確な分割とのずれを表す一例である．更新頻度のピークとなる点から，分割点が正確な一点に集約されている事がわかる（図 2 拡大部）．

また，機能の切り替わる点でない分割点を検出する場合もあったが，これは多くが分割の粒度が極端に小さい時にのみ検出される（図 2 improved method 2）．分割の粒度が適度に大きい場合は機能の切り替わる点が優先的に検出されるため，後述するパラメータ値の設定によって分割の粒度を制御する事で，機能の切り替わる点を正しく反映した分割点のみを得る事ができる．

### 3.2 パラメータによる粒度の変化

各パラメータを単独で変化させた場合，そのパラメータの値を小さくすればするほど，分割によるブロックが細分化され，逆に値を大きくすればブロックが統合されてゆくことが確認できた．

ただし，各パラメータをそれぞれ変化させて分割の粒度を操作した場合は，必ずしも細分化・統合というかたちでブロックが分割されるとは限らない事も分かった．特に，分割の粒度を極端に大きくした場合（実行履歴を二分するなど）は，各パラメータの与え方によって同じ分割粒度でも異なる検出結果となる場合があった．

しかし，分割の粒度が極端に大きい・小さい場合を除くと，分割の粒度が同じであればほぼ共通の機能粒度に即した分割点を検出することができた．

また，上述の結果は履歴上で実現される各機能の規模や偏りによらず，機能粒度と分割粒度を対応させる事ができた．

## 4 まとめと今後の課題

本研究では，実行履歴上に現れるオブジェクト群の変化に着目して実現されている機能の切り替わりを検知する手法を用いて，実際の分割点の決定と分割粒度を制御するための調整を行った．考察で述べた，求められる機能粒度に応じた分割を行うための各パラメータの決定を実行履歴内の情報から自動的に決定する事が今後の課題である．

## 参考文献

[1] 大平，谷口，石尾，神谷，楠本，井上：“ 動作オブジェクト群の変化に着目したオブジェクト指向プログラムの実行履歴分割手法 ” 電子情報通信学会論文誌 D-I, Vol.J88-D-I, No.12, pp.1810-1812, 2005.