Vol.2012-SE-178 No.13
2012/11/2

プログラム実行履歴を用いたコードクローン検出手法

井岡 正和[1,a)]　吉田 則裕[2,b)]　井上 克郎[1,c)]

概要：コードクローンとは，ソースコード中に存在する互いに一致または類似したコード片のことである．コードクローンを検出する手法が，既存研究で数多く提案されている．しかし，既存手法では，プログラムのソースコードやバイナリファイルといった静的な情報のみを使用してコードクローンを検出している．そのため，ソースコードをコピーした後に難読化等の変更が施されると，剽窃が隠蔽されてしまうことがある．そこで，本研究では，プログラムの動的な情報である実行履歴を複数のフェイズに分割し，各フェイズのメソッド呼び出し列を比較することで，コードクローンを検出する手法の提案を行う．提案手法を実際のアプリケーションに適用した結果，難読化前後で同一のコンポーネントを識別できることと，再利用されている箇所を特定できることを確認できた．

キーワード：コードクローン検出，動的解析，難読化

Code Clone Detection Technique Using Program Execution Traces

Masakazu Ioka[1,a)]　Norihiro Yoshida[2,b)]　Katsuro Inoue[1,c)]

Abstract: Code clone is a code fragment that has identical or similar fragments to it in the source code. Many code clone detection techniques and tools have been proposed. However, source code derived by copy-and-paste may be disguised by obfuscation because these techniques detect code clone using only static information such as source code or binary. Therefore, we propose a new clone detection technique, which divides execution trace into a set of phases, and then performs the comparison of those phases based on involved method calls. The experimental result shows that the proposed clone detection technique identified obfuscated and original classes, and an evidence of reusing source code.

Keywords: Code Clone Detection, Dynamic Analysis, Obfuscation

1. はじめに

ソフトウェア工学における研究対象として，コードクローンが注目されている．コードクローンとは，ソースコード中に存在する互いに一致または類似したコード片のことをいい，主にコピーアンドペーストによって生成される．一般的に，コードクローンは，ソフトウェアの保守性を悪化させる要因の 1 つであると考えられている．また，ライセンスを無視したソフトウェアのコピーやソースコードの再利用が年々増加している．このため，コードクローンを自動的に検出する手法が数多く提案されている [3]．

一方で，プログラムの解析を困難にする難読化技術が次々と提案されている [11]．この技術によって，プログラム中に含まれる特定の情報を隠蔽することができる．そのため，ライセンスを無視したソフトウェアのコピーといった剽窃が隠蔽される恐れがある．プログラムに難読化が施されると，多くの既存手法が使用しているプログラムのソースコードやバイナリファイルといった静的な情報のみでは，コードクローンを検出することが難しい．

そこで，本研究では，プログラムの動的な情報である実行履歴からコードクローンを検出する手法を提案する．提

[1] 大阪大学
Osaka University
[2] 奈良先端科学技術大学院大学
Nara Institute of Science and Technology
[a)] m-ioka@ist.osaka-u.ac.jp
[b)] yoshida@is.naist.jp
[c)] inoue@ist.osaka-u.ac.jp

案する手法は，2 つのプログラムの実行履歴を与えると，実行履歴を機能的なまとまりであるフェイズに分割し，フェイズ間の類似度を計算して類似度の高いものをコードクローンとして出力する．

適用例では，実際のアプリケーションに提案手法を適用し，難読化前後で同一のコンポーネントを識別できることと，再利用されている箇所を特定できることを確認した．

以降，2 章では本研究に関連する用語の説明を行う．3 章では提案手法であるプログラム実行履歴からコードクローンを検出する手法について述べ，4 章では実際のアプリケーションに提案手法を適用した結果を述べる．5 章では関連する手法を提案している研究について議論し，最後に 6 章でまとめと今後の課題について述べる．

2. 背景

2.1 コードクローンとその検出

2.1.1 コードクローン

コードクローンとは，ソースコード内の同一または類似するコード片のことをいう [1]．コードクローンは，主に既存コードのコピーアンドペーストや，定型処理の記述によって生成される [6]．一般的に，コードクローンは，ソフトウェアの保守性を悪化させる要因の 1 つであると考えられている．

2.1.2 コードクローン検出手法

コードクローンを検出する手法が，既存研究で数多く提案されている [4], [5], [7], [9], [10], [14]．コードクローン検出手法は，プログラムの品質管理やプログラムの剽窃検出に利用されている．また，これらのコードクローン検出手法は，コードクローンの検出単位によって，大まかに以下の 5 つに分類することができる．

- 行単位の検出
- 字句単位の検出
- 抽象構文木を用いた検出
- プログラム依存グラフを用いた検出
- メトリクスやフィンガープリント等，その他の技術を用いた検出

この分類において，上に記述したものほど高速にコードクローンを検出でき，下に記述したものほど差分を含んだコードクローン等の多様なコードクローンを検出することができる．

ここでは，コードクローン検出手法の例として，字句単位の検出にあたる CCFinder[5] について述べる．CCFinder は，プログラムのソースコード中に存在する極大クローンを検出し，その位置を出力する．この手順は，以下の 4 つのステップからなる．

STEP1(字句解析) ソースコードを字句解析し，トークン列に変換する．

STEP2(変換処理) 実用上意味を持たないコードクローンを取り除くことや，些細な表現上の違いを吸収するためにトークン列を変換する．例えば，変数名は同一のトークンに置換されるので，変数名が違うコード片もコードクローンとして判定することができる．

STEP3(検出処理) トークン列の中から，指定された長さ以上一致している部分をコードクローンとしてすべて検出する．

STEP4(出力整形処理) 検出したコードクローンのソースコード上の位置情報を出力する．

2.2 実行履歴の抽出

プログラムの実行履歴は，実行時のオブジェクトの振る舞いを記録したもので，実行されたイベントが時系列順に並んでいる．イベントは，実行されたオブジェクト間のメッセージ通信イベントであり，実行時刻やメッセージを送信したオブジェクトと受信したオブジェクト，メッセージの内容等の情報を保持している [13]．

プログラムの実行履歴を抽出する方法の 1 つとして Amida[12] がある．Amida のプロファイラ機能は，Java プログラムを動的解析し，実行時のイベントを実行履歴として記録するものである．イベントとしては，メソッドの呼び出し，復帰やフィールドの参照，定義等を検出することができる．

2.3 フェイズ分割手法

渡邊らは，1 つの実行履歴を複数のフェイズに分割する手法を提案している [13]．フェイズとは，実行履歴上から切り出された連続するイベント列のうち，“入出力処理” や “データベースアクセス” 等といった，開発者にとって意味のある処理に対応するものを指す．

オブジェクト指向プログラムは，1 つの機能を実行する際に多数の中間データ用のオブジェクトを生成し，その機能の実行が終了した時点でそれらのオブジェクトの大半を破棄するという性質を持っている [8]．そのため，メソッド呼び出しイベントに関わったオブジェクトを Least Recently Used (LRU) キャッシュに登録していくことで，キャッシュの更新頻度からあるフェイズが終了し次のフェイズが開始したことを検知することができる．

また，このアルゴリズムは入力する実行履歴のサイズにほぼ比例した時間コストで計算可能である．

2.4 難読化技術

難読化とは，あるプログラムを理解が困難な等価なプログラムに変換することである [11]．難読化を施すことによって，プログラムを不正な解析から保護することができる．一方で，不正にコピーしたプログラムに難読化を施すことによって，剽窃が隠蔽される恐れがある．

難読化には様々な技術が存在するが，ここでは，名前変

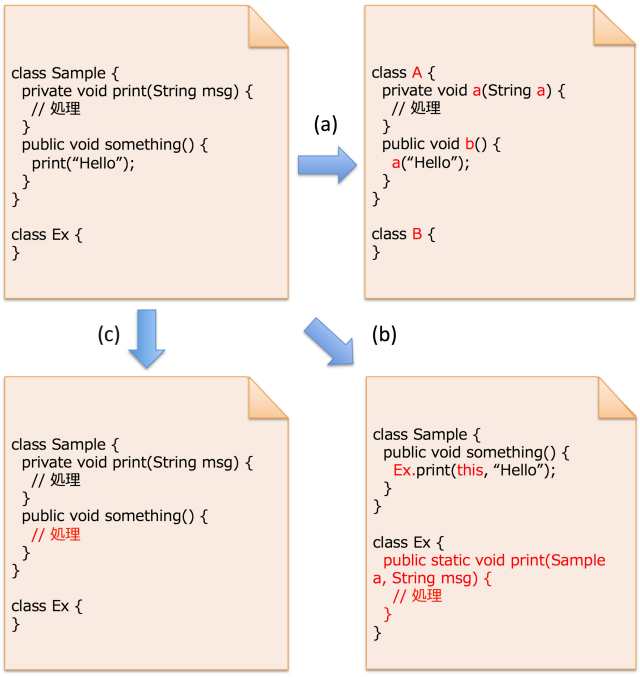

図 1 難読化の例
Fig. 1 An example of obfuscations

換，メソッド分散，インライン展開について述べる．

名前変換 対象クラスに含まれるシンボル名 (クラス名，フィールド名，メソッド名等) の定義を変更し，意味のない名前にする技術である．定義を変更するため，システムが提供する API に含まれる名前を変更することはできない．例を図 1 (a) に示す．例ではシンボル名を a，b 等に変換している．

メソッド分散 対象クラスの集合からランダムに選択したメソッドを別のクラスに移動する技術である．例を図 1 (b) に示す．呼び出されるメソッドの修飾子を public static にし，呼び出す側はこのメソッドを静的に呼び出すように変更している．また，移動したメソッドから移動元クラスのフィールドを参照できるように，引数に移動元のオブジェクトを渡すように変更している．

インライン展開 対象クラスの集合からランダムに選択したメソッド呼び出しにそのメソッドのコードを展開する技術である．例を図 1 (c) に示す．例では，something メソッド内の print メソッドの呼び出しに，print メソッドのコードを展開している．

3. 提案手法

本稿では，2 つのプログラム実行履歴を与えると，その 2 つの実行履歴からコードクローンを検出する手法を提案する．この手法では，プログラムの実行履歴を対象としているため，プログラムを難読化された場合でもコードクローンを検出することができる．なお，プログラムの実行履歴は，イベントとして “メソッドの呼び出し” と “New 演算 (コンストラクタの呼び出し)” を取得したものを与えるものとする．

提案手法の概要を図 2 に示す．提案手法は，以下の手順からなる．

(1) フェイズ分割 与えられたプログラムの実行履歴をフェイズに分割．

(2) 正規化 各フェイズのメソッド呼び出しを正規化．

(3) フェイズ比較 動的計画法を用いて各フェイズを比較．

以降，各手順について詳述する．

3.1 フェイズ分割

与えられたプログラムの実行履歴を，渡邊らのフェイズ分割手法 [13] を用いて複数のフェイズに分割する．このとき，フェイズの長さが短いものは異なる処理のフェイズであってもメソッド呼び出し列が重複しやすいので，フェイズの長さが閾値未満のものを検出対象から除外する．

3.2 正規化

各フェイズについて 2 種類の正規化を行う．

1 つ目は，メソッド呼び出し列の正規化である．メソッド呼び出しの繰り返し回数に意味を持たないことが多いので，2 回以上のメソッド呼び出しは 2 回のメソッド呼び出しとする．

2 つ目は，メソッド呼び出しシグネチャの正規化である．難読化等によってメソッドのシグネチャが意味を持たないことが多いので，メソッド呼び出し内の出現位置情報を使ってインデックスを振る．メソッド呼び出しの正規化は以下の 2 つの方法で行う．

正規化 1 (1 つの呼び出し内だけで正規化) 1 つのメソッド呼び出し内だけでインデックスを振る．例を図 3 に示す．「メソッド A(型 X, 型 Y); メソッド B(型 Z, 型 Z); メソッド C(型 W, 型 Z);」のそれぞれの呼び出しでインデックスを振るので，「0(1, 2); 0(1, 1); 0(1, 2)」となる．

正規化 2 (直前の呼び出しと現在の呼び出しで正規化) あるメソッド呼び出しを正規化する際，直前のメソッド呼び出しを含めてインデックスを振る．例を図 4 に示す．メソッド A の呼び出しについては，直前にメソッド呼び出しがないので，メソッド A の呼び出し内だけでインデックスを振り「0(1, 2);」となる．メソッド B の呼び出しについては，直前のメソッド呼び出しであるメソッド A を含めてインデックスを振るので，メソッド B の呼び出しは「3(4, 4);」となる．メソッド C の呼び出しについても同様で「2(3, 1);」となる．

正規化 1 では，不要なメソッド呼び出しの追加等のメソッド呼び出しの改変の影響をあまり受けないが，誤検出が多くなる恐れがある．正規化 2 では，メソッド呼び出しの改変の影響を少し受けるが，正規化 1 よりも誤検出が少ない．

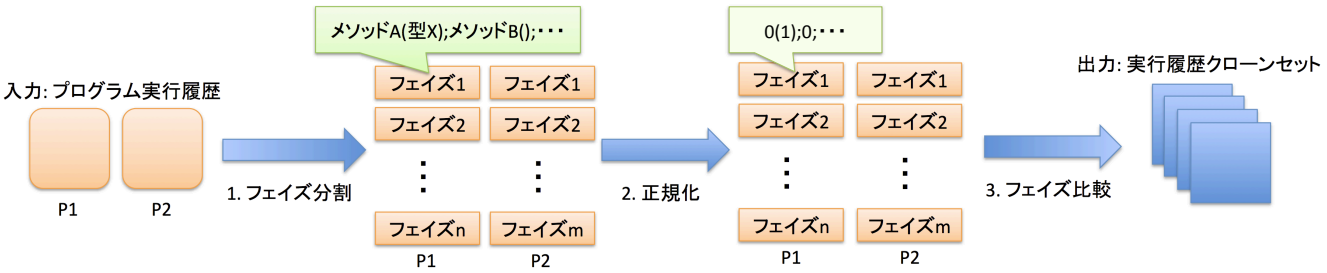

図 2 提案手法の概要
Fig. 2 An overview of the proposed approach

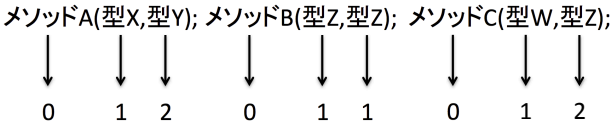

図 3 1 つのメソッド呼び出し内だけで正規化の例
Fig. 3 An example of normalization within one method call

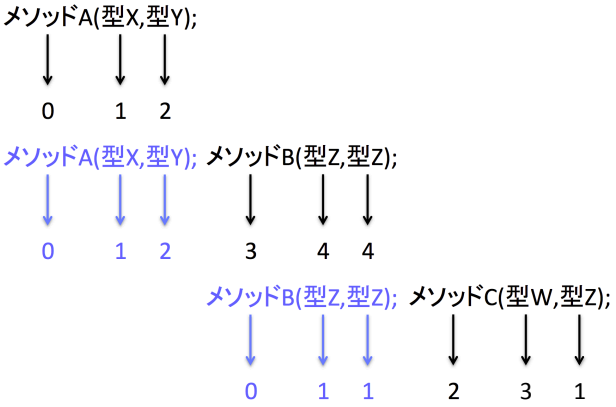

図 4 直前の呼び出しと現在の呼び出しで正規化の例
Fig. 4 An example of normalization according to current and previous method calls

3.3 フェイズ比較

動的計画法を用いた類似文字列マッチングアルゴリズム [15] を使用してフェイズの比較を行う．入力として 2 つの文字列を与え，一方の文字列に，1 文字削除，1 文字挿入，1 文字置換という 3 つの操作を最低何回行ってもう一方と同一の文字列へと変化させられるかを調べることで，2 つの文字列の類似度を求めることができる．

本手法で使用するために，1 メソッド呼び出しを 1 文字に対応付けてこのアルゴリズムを適用する．このアルゴリズムを用いた場合，フェイズ間のメソッド呼び出しの対応関係を得ることができる．

フェイズの比較を行った後，フェイズ間の類似度が高いものからグリーディに対応付けていき，クローンとして出力する．

4. 適用例

提案手法の有効性を確認するために，提案手法を実装し実際のアプリケーションに対して適用した．なお，フェイズの長さの閾値は 50 とし，メソッド呼び出しの正規化は正規化 2 を適用した．適用対象は ICCA[*1]の Gemini コンポーネントと Virgo コンポーネントとし，以下の 2 点を調査の目的とした．

- 難読化前後で同一のコンポーネントを識別できるか．
- 再利用されている箇所を特定できるか．

なお，難読化には，ProGuard[*2]を標準設定で使用した．また，各コンポーネントについて，起動からクローンデータの解析完了までの実行履歴を取得した．

4.1 クラス・メソッド単位のクローン

3.3 節で述べたフェイズの比較によって，フェイズ間でメソッド呼び出しが対応付いている．この対応付いたメソッド呼び出しから，クラス A とクラス B の類似度を以下の式により定義する．

$$similarity(\mathrm{A},\mathrm{B}) = \frac{2 \times (\text{A と B が対応付いているメソッド呼出数})}{\text{A 内のメソッド呼出数} + \text{B 内のメソッド呼出数}}$$

また，この式のクラス A，クラス B をメソッド A，メソッド B に置き換えた式をメソッド間の類似度として定義する．

Gemini コンポーネント，Virgo コンポーネントをそれぞれ ProGuard を用いて難読化し，オリジナルとのクラス間の類似度，メソッド間の類似度を計算した．結果を図 5，6 に示す．このグラフは，類似度に応じたクラス数，メソッド数の累積率を表している．また，Gemini コンポーネントの難読化前後のクラス間類似度に関するヒートマップを図 7 に示す．この結果より，Gemini コンポーネント，Virgo コンポーネント共に，難読化を施してもクラス，メソッドをオリジナルと対応付けることができていることが分かる．

Virgo コンポーネントについて，対応付けが正しいかどうかを確認したところ，すべての対応付けが正しいことが分かった．また，クラス間やメソッド間の類似度があまり

*1 ICCA: http://sel.ist.osaka-u.ac.jp/icca/
*2 ProGuard: http://proguard.sourceforge.net/

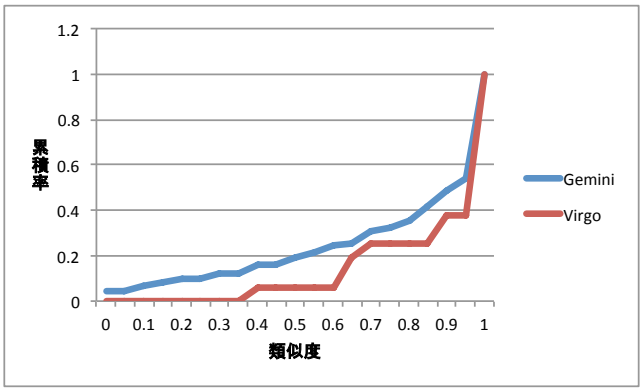

図 5 難読化前後のクラス間類似度
Fig. 5 Similarity between obfuscated and original classes

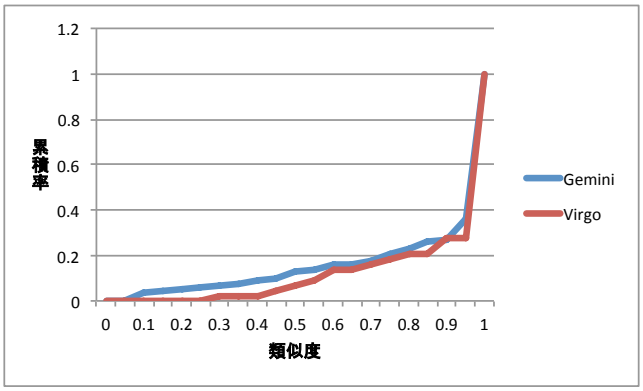

図 6 難読化前後のメソッド間類似度
Fig. 6 Similarity between obfuscated and original methods

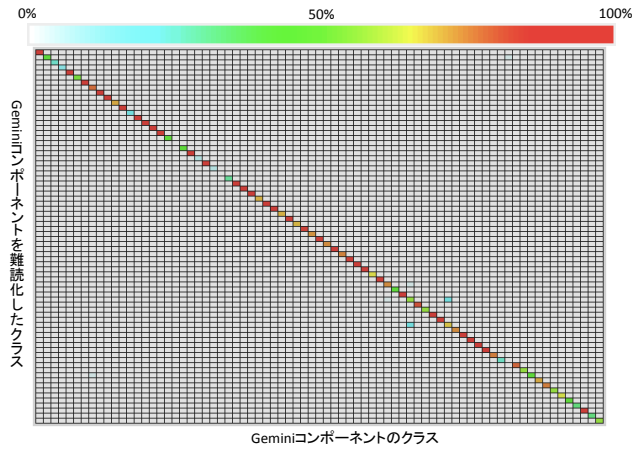

図 7 難読化前後のクラス間類似度に関するヒートマップ
Fig. 7 Heat map of similarity between obfuscated and original classes

高くないものが存在するのは，難読化によってメソッドがインライン展開され，メソッド呼び出しの構造が変化したことが原因であると分かった．

このことより，難読化前後で同一のコンポーネントを識別できることを確認できた．

4.2 類似したフェイズの調査

Gemini コンポーネントと Virgo コンポーネントに対して，類似したフェイズがどのような特徴を持っているか調査した．調査した組み合わせは，Gemini コンポーネントと Virgo コンポーネント (ケース 1)，Gemini コンポーネントと Virgo コンポーネントを難読化したもの (ケース 2)，Gemini コンポーネントを難読化したものと Virgo コンポーネント (ケース 3)，の 3 つのケースである．

各ケースの類似フェイズの検出結果を表 1 に示す．検出数は，ケース 3 のみ少なくなっている．これは，ProGuard による難読化では各フェイズの長さがオリジナルよりも短くなるという特徴があり，規模の大きな Gemini コンポーネントは影響を受けやすかったためであると考えられる．最大類似度については，すべてのケースで 1.00 となっている．Gemini コンポーネントと Virgo コンポーネントでは，共通のライブラリである iccalib を使用している．このライブラリ内のメソッドで始まるフェイズが完全に一致していたため，類似度が 1.00 となっていた．一方，最小類似度については，ケース 3 が非常に小さい値となっている．これは，Gemini コンポーネントには多くのメソッドが存在し，ProGuard による難読化によって，メソッド呼び出しの構造の変化が大きかったためであると考えられる．最後に，平均類似度については，ケース 3 の値が少し高くなっているがほぼ同じ値となった．

また，iccalib 内のメソッド以外で始まるフェイズについて，どのようなフェイズが類似しているかを確認した．ケース 2 における類似フェイズで，Gemini コンポーネント内のフェイズではファイルからクローン情報を取得して解析しており，また，Virgo コンポーネント内のフェイズではクローンセットからクローン情報を取得して解析していた．Virgo コンポーネントは，Gemini コンポーネントより後に開発されているので，Gemini コンポーネントのプログラムを再利用して Virgo コンポーネントを作成したと考えられる．この類似フェイズは付録に添付している．

以上のことより，再利用されている箇所を特定できることを確認できた．

5. 関連研究

Sæbjørnsen らは，バイナリファイルからコードクローンを検出する手法を提案している [10]．この手法は，バイナリをディスアセンブルし，そのアセンブラの特徴を比較

表 1 類似フェイズの検出結果
Table 1 Detection result of similar phases

	検出数	最大類似度	最小類似度	平均類似度
ケース 1	75	1.00	0.227	0.612
ケース 2	75	1.00	0.192	0.612
ケース 3	61	1.00	0.048	0.632

してコードクローンを検出する．そのため，ソースコードに対して行うコードクローン検出では検出できないコードクローンを検出することができる．しかし，逆に見つけることができないコードクローンも存在する．

Lim らは，制御フローグラフを用いた静的な Java プログラムの盗用検出手法を提案している [9]．この手法では，Java バイトコードから制御フローグラフを作成し，プログラムが実行しうるパスを比較してコードクローンを検出している．そのため，単純なソースコードの比較やバイナリの比較では検出できなかったコーロクローンを検出することができる．また，静的な情報のみを使用しているので，プログラム全体をカバーすることが容易である．

これらの既存手法と比較すると，提案手法は，より実際の振る舞いを考慮したコーロクローン検出ができるという利点がある．しかし，提案手法では，プログラムの実行履歴を使用するため，実行環境の構築 (プログラムへの入力等) が必要となり，プログラム全体を対象にしたコードクローン検出が困難である．

また，提案手法と同じプログラムの実行履歴を比較して類似した箇所を可視化する手法を，Cornelissen らが提案している [2]．この手法は，プログラムの実行履歴の新しい見識を得てプログラム理解を容易にすることを目的としており，2 つの実行履歴を比較して，呼び出し元，呼び出し先，シグネチャ，ランタイムパラメータのすべてが同じものを一致とみなし，その結果をスキャッタープロットで可視化している．一方，提案手法では，2 つの実行履歴を機能的なまとまりであるフェイズに分割，正規化した後，各フェイズを動的計画法を用いて比較し，クラスのクローン，メソッドのクローンをヒートマップで可視化している．

6. まとめと今後の課題

本稿では，プログラムの実行履歴をフェイズに分割し，各フェイズのメソッド呼び出し列を比較することで，コードクローンを検出する手法を提案した．そして，実際のアプリケーションに提案手法を適用して，その有効性を確認した．

今後の課題として，まず，メソッド呼び出しの正規化を改善することが挙げられる．メソッド呼び出しの正規化を改善することによって，複数メソッドにまたがるクローンの検出や，難読化によるインライン展開等のメソッド呼び出し構造の変化に対応することが可能となる．また，提案手法を定量的に評価することも今後の課題である．

謝辞 本研究は，日本学術振興会科学研究費補助金基盤研究 (A) (課題番号: 21240002)，基盤研究 (C) (課題番号: 22500026) の助成を得た．

参考文献

[1] I. D. Baxter, A. Yahin, L. Moura, M. S. Anna, and L. Bier. Clone Detection Using Abstract Syntax Trees. In *Proc. of ICSM 1998*, pp. 368–377, 1998.

[2] B. Cornelissen and L. Moonen. Visualizing similarities in execution traces. In *Proc. of PCODA 2007*, pp. 6–10, 2007.

[3] 肥後芳樹, 楠本真二, 井上克郎. コードクローン検出とその関連技術. 電子情報通信学会論文誌, Vol. J91-D, No. 6, pp. 1465–1481, 2008.

[4] L. Jiang, G. Misherghi, Z. Su, and S. Glondu. DECKARD: Scalable and accurate tree-based detection of code clones. In *Proc. of ICSE 2007*, pp. 96–105, 2007.

[5] T. Kamiya, S. Kusumoto, and K. Inoue. CCFinder: A multilinguistic token-based code clone detection system for large scale source code. *IEEE Trans. Softw. Eng.*, Vol. 28, No. 7, pp. 654–670, 2002.

[6] M. Kim, L. Bergman, T. Lau, and D. Notkin. An Ethnographic Study of Copy and Paste Programming Practices in OOPL. In *Proc. of ISESE 2004*, pp. 83–92, 2004.

[7] J. Krinke. Identifying similar code with program dependence graphs. In *Proc. of WCRE 2001*, pp. 301–309, 2001.

[8] H. Lieberman and C. Hewitt. A real-time garbage collector based on the lifetimes of objects. *Communications of the ACM*, Vol. 26, No. 6, pp. 419–429, 1983.

[9] H. Lim, H. Park, S. Choi, and T. Han. A method for detecting the theft of Java programs through analysis of the control flow information. *Information and Software Technology*, Vol. 51, No. 9, pp. 1338–1350, 2009.

[10] A. Sæbjørnsen, J. Willcock, T. Panas, D. Quinlan, and Z. Su. Detecting code clones in binary executables. In *Proc. of ISSTA 2009*, pp. 117–128, 2009.

[11] 玉田春昭, 中村匡秀, 門田暁人, 松本健一. Java クラスファイル難読化ツール DonQuixote. 日本ソフトウェア科学会 FOSE 2006, pp. 113–118, 2006.

[12] 谷口考治, 石尾隆, 神谷年洋, 楠本真二, 井上克郎. プログラム実行履歴からの簡潔なシーケンス図の生成手法. コンピュータソフトウェア, Vol. 24, No. 3, pp. 153–169, 2007.

[13] 渡邊結, 石尾隆, 井上克郎. 協調動作するオブジェクト群の変化に基づく実行履歴の自動分割. 情報処理学会論文誌, Vol. 51, No. 12, pp. 2273–2286, 2010.

[14] R. Wettel and R. Marinescu. Archeology of code duplication: recovering duplication chanins from small duplication fragments. In *Proc. of SYNASC 2005*, pp. 63–70, 2005.

[15] R. B. Yates and B. R. Neto. *Modern Information Retrieval*. Addison Wesley, 1999.

付　　録

A.1 類似フェイズ

4.2 節で述べた，再利用されているフェイズを図 A·1 に示す．

Vol.2012-SE-178 No.13
2012/11/2

Geminiコンポーネント	Virgoコンポーネントの難読化
gemini.view.visual.scatterplot.ScatterPlotPanel (gemini.view.visual.scatterplot.ScatterPlotPanel.drawGroupSeparators)	virgo.a.a(iccalib.data.file.FileInfo.getNOC)
gemini.view.visual.scatterplot.ScatterPlotPanel (gemini.view.visual.scatterplot.ScatterPlotPanel.getScaleRatio)	
gemini.view.visual.scatterplot.ScatterPlotPanel(gemini.data.file.IDIndexMap.getFile)	iccalib.data.file.FileInfo(iccalib.data.file.FileMetricsManager.getNOC)
gemini.view.visual.scatterplot.ScatterPlotPanel(gemini.data.file.IDIndexMap.getFile)	iccalib.data.file.FileMetricsManager(iccalib.data.file.FileMetricsManager.getCodeFragments)
gemini.view.visual.scatterplot.ScatterPlotPanel(gemini.data.file.FileOffsetData.get)	iccalib.data.file.FileMetricsManager(iccalib.data.file.FileMetricsManager.getCodeFragments)
iccalib.data.file.FileInfo(iccalib.data.file.FileInfo.getPath)	iccalib.data.file.FileInfo(iccalib.data.file.FileInfo.getPath)
gemini.view.visual.scatterplot.ScatterPlotPanel(gemini.data.file.FileOffsetData.get)	iccalib.data.file.FileMetricsManager(iccalib.data.file.FileMetricsManager.createCodeFragments)
iccalib.data.file.FileInfo(iccalib.data.file.FileInfo.getPath)	iccalib.data.file.FileMetricsManager(iccalib.data.file.FileInfo.getGroupID)
gemini.view.visual.scatterplot.ScatterPlotPanel(iccalib.data.file.FileInfo.getTOC)	iccalib.data.file.FileMetricsManager(iccalib.data.file.FileInfo.getFileID)
gemini.view.visual.scatterplot.ScatterPlotPanel (gemini.view.visual.scatterplot.ScatterPlotPanel.getLowerLength)	iccalib.data.file.FileMetricsManager(iccalib.data.clone.CloneManager.getFileCodeFragments)
…	…
gemini.view.visual.scatterplot.ScatterPlotPanel(gemini.data.file.FileOffsetData.get)	iccalib.data.clone.CodeFragmentInfo(iccalib.data.clone.CodeFragmentInfo.compareTo)
iccalib.data.file.FileInfo(iccalib.data.file.FileInfo.getPath)	iccalib.data.clone.CodeFragmentInfo(iccalib.data.clone.CodeFragmentInfo.getGroupID)
gemini.view.visual.scatterplot.ScatterPlotPanel(iccalib.data.file.FileManager.getLastFile)	
…	
iccalib.data.file.FileManager$FileKey(iccalib.data.file.FileManager$FileKey.getGroupID)	
iccalib.data.file.FileManager$FileKey(iccalib.data.file.FileManager$FileKey.getGroupID)	iccalib.data.clone.CodeFragmentInfo(iccalib.data.clone.CodeFragmentInfo.getGroupID)
iccalib.data.file.FileManager$FileKey(iccalib.data.file.FileManager$FileKey.getFileID)	iccalib.data.clone.CodeFragmentInfo(iccalib.data.clone.CodeFragmentInfo.getFileID)
iccalib.data.file.FileManager$FileKey(iccalib.data.file.FileManager$FileKey.getFileID)	iccalib.data.clone.CodeFragmentInfo(iccalib.data.clone.CodeFragmentInfo.getFileID)
iccalib.data.file.FileManager$FileKey(iccalib.data.file.FileManager$FileKey.compareTo)	iccalib.data.clone.CodeFragmentInfo(iccalib.data.clone.CodeFragmentInfo.compareTo)
iccalib.data.file.FileManager$FileKey(iccalib.data.file.FileManager$FileKey.getGroupID)	iccalib.data.clone.CodeFragmentInfo(iccalib.data.clone.CodeFragmentInfo.getGroupID)
iccalib.data.file.FileManager$FileKey(iccalib.data.file.FileManager$FileKey.getGroupID)	iccalib.data.clone.CodeFragmentInfo(iccalib.data.clone.CodeFragmentInfo.getGroupID)
iccalib.data.file.FileManager$FileKey(iccalib.data.file.FileManager$FileKey.compareTo)	iccalib.data.clone.CodeFragmentInfo(iccalib.data.clone.CodeFragmentInfo.getFileID)
iccalib.data.file.FileManager$FileKey(iccalib.data.file.FileManager$FileKey.getGroupID)	
iccalib.data.file.FileManager$FileKey(iccalib.data.file.FileManager$FileKey.getGroupID)	iccalib.data.clone.CodeFragmentInfo(iccalib.data.clone.CodeFragmentInfo.getFileID)
iccalib.data.file.FileManager$FileKey(iccalib.data.file.FileManager$FileKey.compareTo)	iccalib.data.clone.CodeFragmentInfo(iccalib.data.clone.CodeFragmentInfo.compareTo)
iccalib.data.file.FileManager$FileKey(iccalib.data.file.FileManager$FileKey.getGroupID)	iccalib.data.clone.CodeFragmentInfo(iccalib.data.clone.CodeFragmentInfo.getGroupID)
iccalib.data.file.FileManager$FileKey(iccalib.data.file.FileManager$FileKey.getGroupID)	iccalib.data.clone.CodeFragmentInfo(iccalib.data.clone.CodeFragmentInfo.getGroupID)
iccalib.data.file.FileManager$FileKey(iccalib.data.file.FileManager$FileKey.compareTo)	
iccalib.data.file.FileManager$FileKey(iccalib.data.file.FileManager$FileKey.getGroupID)	
iccalib.data.file.FileManager$FileKey(iccalib.data.file.FileManager$FileKey.getGroupID)	
iccalib.data.file.FileManager(iccalib.data.file.FileManager$FileKey.getFileID)	iccalib.data.clone.CodeFragmentInfo(iccalib.data.clone.CodeFragmentInfo.getFileID)
iccalib.data.file.FileManager(iccalib.data.file.FileManager.getFile)	iccalib.data.clone.CodeFragmentInfo(iccalib.data.clone.CodeFragmentInfo.getFileID)
iccalib.data.file.FileManager(iccalib.data.file.FileManager$FileKey.)	iccalib.data.clone.CodeFragmentInfo(iccalib.data.clone.CodeFragmentInfo.compareTo)
iccalib.data.file.FileManager$FileKey(iccalib.data.file.FileManager$FileKey.compareTo)	
iccalib.data.file.FileManager$FileKey(iccalib.data.file.FileManager$FileKey.getGroupID)	iccalib.data.clone.CodeFragmentInfo(iccalib.data.clone.CodeFragmentInfo.getGroupID)
iccalib.data.file.FileManager$FileKey(iccalib.data.file.FileManager$FileKey.getGroupID)	iccalib.data.clone.CodeFragmentInfo(iccalib.data.clone.CodeFragmentInfo.getGroupID)
iccalib.data.file.FileManager$FileKey(iccalib.data.file.FileManager$FileKey.getFileID)	iccalib.data.clone.CodeFragmentInfo(iccalib.data.clone.CodeFragmentInfo.getFileID)
iccalib.data.file.FileManager$FileKey(iccalib.data.file.FileManager$FileKey.getFileID)	iccalib.data.clone.CodeFragmentInfo(iccalib.data.clone.CodeFragmentInfo.getFileID)
iccalib.data.file.FileManager$FileKey(iccalib.data.file.FileManager$FileKey.compareTo)	iccalib.data.clone.CodeFragmentInfo(iccalib.data.clone.CodeFragmentInfo.compareTo)
iccalib.data.file.FileManager$FileKey(iccalib.data.file.FileManager$FileKey.getGroupID)	iccalib.data.clone.CodeFragmentInfo(iccalib.data.clone.CodeFragmentInfo.getGroupID)
iccalib.data.file.FileManager$FileKey(iccalib.data.file.FileManager$FileKey.getGroupID)	iccalib.data.clone.CodeFragmentInfo(iccalib.data.clone.CodeFragmentInfo.getGroupID)
iccalib.data.file.FileManager$FileKey(iccalib.data.file.FileManager$FileKey.getFileID)	iccalib.data.clone.CodeFragmentInfo(iccalib.data.clone.CodeFragmentInfo.getFileID)
iccalib.data.file.FileManager$FileKey(iccalib.data.file.FileManager$FileKey.getFileID)	iccalib.data.clone.CodeFragmentInfo(iccalib.data.clone.CodeFragmentInfo.getFileID)
iccalib.data.file.FileManager$FileKey(iccalib.data.file.FileManager$FileKey.compareTo)	iccalib.data.clone.CodeFragmentInfo(iccalib.data.clone.CodeFragmentInfo.compareTo)
iccalib.data.file.FileManager$FileKey(iccalib.data.file.FileManager$FileKey.getGroupID)	iccalib.data.clone.CodeFragmentInfo(iccalib.data.clone.CodeFragmentInfo.getGroupID)
iccalib.data.file.FileManager$FileKey(iccalib.data.file.FileManager$FileKey.getGroupID)	iccalib.data.clone.CodeFragmentInfo(iccalib.data.clone.CodeFragmentInfo.getGroupID)
iccalib.data.file.FileManager$FileKey(iccalib.data.file.FileManager$FileKey.getFileID)	iccalib.data.clone.CodeFragmentInfo(iccalib.data.clone.CodeFragmentInfo.getFileID)
iccalib.data.file.FileManager$FileKey(iccalib.data.file.FileManager$FileKey.getFileID)	iccalib.data.clone.CodeFragmentInfo(iccalib.data.clone.CodeFragmentInfo.getFileID)
iccalib.data.file.FileManager$FileKey(iccalib.data.file.FileManager$FileKey.compareTo)	iccalib.data.clone.CodeFragmentInfo(iccalib.data.clone.CodeFragmentInfo.compareTo)
iccalib.data.file.FileManager$FileKey(iccalib.data.file.FileManager$FileKey.getGroupID)	iccalib.data.clone.CodeFragmentInfo(iccalib.data.clone.CodeFragmentInfo.getGroupID)
iccalib.data.file.FileManager$FileKey(iccalib.data.file.FileManager$FileKey.getGroupID)	iccalib.data.clone.CodeFragmentInfo(iccalib.data.clone.CodeFragmentInfo.getGroupID)
iccalib.data.file.FileManager$FileKey(iccalib.data.file.FileManager$FileKey.getFileID)	iccalib.data.clone.CodeFragmentInfo(iccalib.data.clone.CodeFragmentInfo.getFileID)
iccalib.data.file.FileManager$FileKey(iccalib.data.file.FileManager$FileKey.getFileID)	iccalib.data.clone.CodeFragmentInfo(iccalib.data.clone.CodeFragmentInfo.getFileID)
gemini.view.visual.scatterplot.ScatterPlotPanel(iccalib.data.file.FileInfo.getGroupID)	iccalib.data.clone.CodeFragmentInfo(iccalib.data.clone.CodeFragmentInfo.compareTo)
…	
iccalib.data.file.FileManager$FileKey(iccalib.data.file.FileManager$FileKey.compareTo)	
iccalib.data.file.FileManager$FileKey(iccalib.data.file.FileManager$FileKey.getGroupID)	iccalib.data.clone.CodeFragmentInfo(iccalib.data.clone.CodeFragmentInfo.getGroupID)
iccalib.data.file.FileManager$FileKey(iccalib.data.file.FileManager$FileKey.getGroupID)	iccalib.data.clone.CodeFragmentInfo(iccalib.data.clone.CodeFragmentInfo.getGroupID)
iccalib.data.file.FileManager$FileKey(iccalib.data.file.FileManager$FileKey.getFileID)	iccalib.data.clone.CodeFragmentInfo(iccalib.data.clone.CodeFragmentInfo.getFileID)
iccalib.data.file.FileManager$FileKey(iccalib.data.file.FileManager$FileKey.getFileID)	iccalib.data.clone.CodeFragmentInfo(iccalib.data.clone.CodeFragmentInfo.getFileID)
iccalib.data.file.FileManager$FileKey(iccalib.data.file.FileManager$FileKey.compareTo)	iccalib.data.clone.CodeFragmentInfo(iccalib.data.clone.CodeFragmentInfo.compareTo)
…	…
iccalib.data.file.FileManager$FileKey(iccalib.data.file.FileManager$FileKey.getGroupID)	iccalib.data.clone.CodeFragmentInfo(iccalib.data.clone.CodeFragmentInfo.getGroupID)
iccalib.data.file.FileManager$FileKey(iccalib.data.file.FileManager$FileKey.getGroupID)	iccalib.data.clone.CodeFragmentInfo(iccalib.data.clone.CodeFragmentInfo.getGroupID)
iccalib.data.file.FileManager$FileKey(iccalib.data.file.FileManager$FileKey.getFileID)	iccalib.data.clone.CodeFragmentInfo(iccalib.data.clone.CodeFragmentInfo.getFileID)
iccalib.data.file.FileManager$FileKey(iccalib.data.file.FileManager$FileKey.getFileID)	iccalib.data.clone.CodeFragmentInfo(iccalib.data.clone.CodeFragmentInfo.getFileID)
iccalib.data.file.FileInfo(iccalib.data.file.FileInfo.getPath)	iccalib.data.file.FileInfo(iccalib.data.file.FileInfo.getPath)

図 A·1 検出類似フェイズの例

Fig. A·1 An example of detected pair of similar phases