Java バイトコードの比較による機能的類似コード検出手法の提案

2020SE038 宮田千里
指導教員：名倉正剛

1　はじめに

ソフトウェア開発において，類似するコードの存在は，プログラムの保守性を低下させる．したがって，類似コードを検出し，削減することでプログラムの保守性を向上させることができる．類似コードは，コピー&ペーストなどによって発生する場合や開発者が意図せずに作成してしまうことで発生する場合がある．前者はプログラムの構文や構造や識別子が類似することが多く，そのような類似コードを検出する手法が多く提案されている．一方で後者はプログラムの機能は類似するが，構文や構造や識別子が類似せず，そのような類似コードの検出は困難である．本研究では，このようなコードを機能的類似コードとして定義し，検出する手法を提案する．

2　機能的類似コード

複数人による並行開発などで同じような機能を別々に実装してしまうなど，コピー&ペーストに起因しない類似コードが存在する [1]．この場合，コード断片の機能は類似するが，構造や構文や識別子が類似しないことが多く，既存の類似コード検出手法では検出が困難である．本研究では，機能が複数の処理によって構成されるという前提をもとに，機能を構成する処理が類似するコード断片を機能的類似コードとして定義する．そして，機能の類似性に着目した類似コード検出を行うことで，構造や構文や識別子が類似しない類似コードについても検出を試みる．

3　提案手法

3.1　概要

本研究では，Java ソースコードをコンパイルした結果として得られる機械語命令を比較することによって，機能的類似コードを検出する手法を提案する．機能的類似コードとなる複数のコード片では，構造や構文が異なったとしても内部的に同じような計算が行われ，そうであればプログラムに含む機械語命令のセットも類似する．そこで本研究では，機能的類似コードとなる複数のコード片では Java バイトコードの命令セットが類似すると考え，比較対象のコード片に対応するバイトコードに含むオペコードとその出現回数を比較することで，類似コード片を検出する．

3.2　Java バイトコード

Java のソースコードをコンパイルした際に生成されるクラスファイルには，Java 仮想マシンで実行される中間コード（バイトコード）が記述される．バイトコードには仮想マシンで動作する命令列が，操作の種類を表す符号（オペコード）と操作対象（オペランド）の組合せのセットとして表現される．このオペコードは，1 バイト（256 種類，ただし 51 個は未定義）で構成される．

3.3　処理の流れ

提案手法の構成と処理の流れを，図 1 に示す．処理の流れは次のとおりである．

手順 1) バイトコードの命令列の抽出
比較対象のコード片を含むクラスファイルを解析し，該当のコード片に対応するバイトコードに含む命令列を抽出する．

手順 2) オペコードごとの出現回数の計測
抽出したバイトコードの命令列から，オペコードごとに出現回数を計測する．

手順 3) メソッド呼び出しの統合
命令列内にメソッド呼び出しが存在する場合，呼び出し先メソッドの命令列についてもオペコードごとの計測を行う．

手順 4) コード片間の類似度の算出
オペコードごとの出現回数の相違を，コード片間の距離（これを「機能的距離」と呼ぶ）として算出し，機能的距離を利用して類似度を算出する．

手順 2 では，同じ振る舞いの命令を，同じ命令として計測する．Java バイトコードでは，同じ振る舞いでも，オペランドによって異なるオペコードが用意されている．そのため，全てのオペコードを操作の種類として定義すると，同じ振る舞いの命令を別の操作の種類として計測してしまう．提案手法では，オペコードを振る舞いで分類し，同じ振る舞いの命令を同じ命令として計測することで，オペランドによる命令の差異を除去している．

手順 3 では，抽出した命令列にメソッド呼び出しが含まれる場合，呼び出されるメソッドの命令列を呼び出し元に統合する．Java バイトコードでは，呼び出し先で実施される処理が，呼び出し元の命令列内で表現されないため，オペコードとその出現回数がメソッド分割の粒度に依存する．したがって，メソッド呼び出しを統合することで，メソッド分割の粒度が異なる場合の影響を排除する．ただし，外部メソッドの類似性は考慮しないため，プロジェクト内のメソッドの呼び出しのみ統合する．

手順 4 では，比較対象のコード片間で類似度の高いコード片同士を類似コードとして判定する．本章の残りでは，手順 3 における類似度の算出方法について述べる．

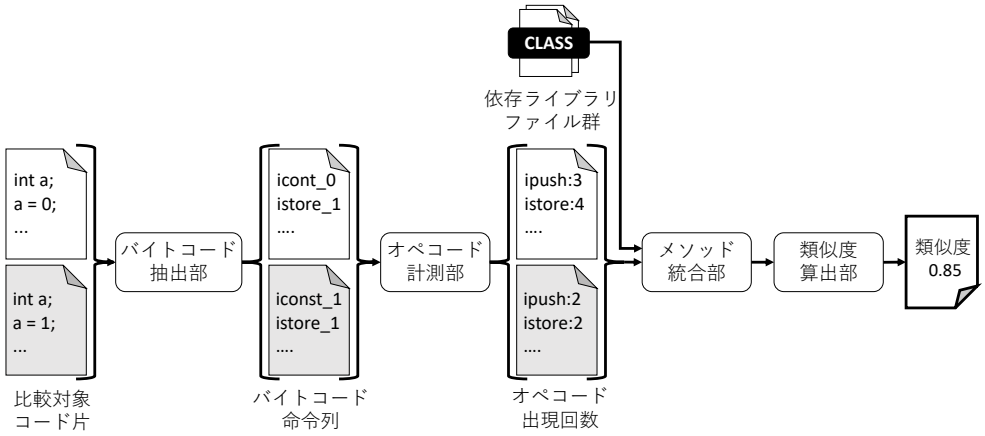

図 1　提案手法の流れ

表 1　類似度の分布

類似度	ペア数	割合
100	432	0.322
90 ∼	323	0.241
80 ∼	310	0.231
70 ∼	151	0.113
60 ∼	59	0.044
50 ∼	37	0.028
40 ∼	12	0.009
30 ∼	7	0.005
20 ∼	5	0.004
10 ∼	4	0.003
0 ∼	2	0.001
合計	1342	

3.4　類似度の算出方法

手順 4 では，まず，(1) 式に示す定義により機能的距離を算出する．

$$Dist(A,B) = \sum_{i=1}^{n} |OpA_i - OpB_i| \qquad (1)$$

(1) 式では，2 つのコード片 A と B に対して，オペコードの種類ごとの出現回数の差分を合計した値を機能的距離 $Dist(A,B)$ として算出する．OpA, OpB はそれぞれコード片 A,B の全オペコード数であり，OpA_i, OpB_i はそれぞれコード片 A,B におけるオペコード i の出現回数を示す．

次に，(2) 式に示す定義により類似度を算出する．

$$Sim(A,B) = 1 - \frac{Dist(A,B)}{OpA + OpB} \qquad (2)$$

コード片 A,B の全オペコード数が相違していた場合の機能的距離は，コード片 A と B の命令数を加算した値が機能的距離となる．全ての命令の種類が相違していた場合を 1 とした場合の相違の割合を 1 から引いた値を類似度として定義する．この類似度は，2 つのコード片に存在する全ての命令数に対して，共通して存在する命令の数の割合を示している．

4　評価実験

提案手法が機能的類似コードを検出できることを確認する目的で，機能が同一なメソッドを収集したデータセット [2] を用いて，それぞれのメソッド間の類似度を算出して比較した．このデータセット [2] は，テストケース実行時の入出力が同一なメソッドを，機能等価メソッドとして定義し，収集したものである．機能等価メソッドに対して，提案手法で類似度を算出した際の分布を表 1 に示す．表 1 にあるように，機能等価メソッドの組に対して提案手法では，9 割程度が 70 %以上の類似度を示した．

また，提案手法が実際に機能的類似コードを検出できるかを確認する目的で，OSS プロジェクトを対象に機能的類似コードを検出し比較した．その結果，機能的類似コードが検出できていることが確認できた．

5　関連研究

機能的類似コードの検出方法の一つとして入出力に着目した方法がある [2]．この方法ではテストケース実行時の入出力が一致しているならば同一とみなすことで機能的類似コードを検出する．その際，機能的に類似しているコードの範囲がメソッドであるということを前提に，メソッドの入出力の類似性によって機能的類似コードを検出している．しかし，実際にはメソッド内部の部分的なコード片が類似することもあり，メソッドの入出力が類似するとは限らない．そのため，本研究では，入出力や構文や構造に関わらず，類似部分が含まれているコード片を検出対象としている．

6　まとめ

本研究では，Java バイトコードを比較することで，機能的類似コードを検出する手法を提案した．そして，評価実験を通じて提案手法で機能的類似コードを検出できる可能性を示した．既存のプロジェクトに対して，提案手法を利用することで機能的類似コードを検出し，リファクタリングや重複コードの削減を行うことができる．

参考文献

[1] V. Käfer, S. Wagner, and R. Koschke. Are there functionally similar code clones in practice? In *12th Int'l Workshop on Soft. Clones (IWSC 2018)*, pp. 2–8, 2018.

[2] 肥後芳樹．自動テスト生成技術を利用した機能等価メソッドデータセットの構築．ソフトウェアエンジニアリングシンポジウム 2023(SES2023), pp. 30–38, 2023.