コード片が利用するメソッドの類似性に基づく機能的類似コード検出手法の提案

2020SE044 中村伊吹
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，類似コードが存在すると保守性が低下する．プロジェクト内に類似コードが存在すると，プログラム変更時に複数箇所での修正が必要となり，変更コストの増大や修正漏れのリスクが高くなる．類似コードは一般的にソースコードのコピー&ペーストによって発生することが多いが，異なる開発者が同じ機能を独立して実装することで発生する場合もある．特に後者の場合は，ソースコード中の識別子名や構文や制御構造などのプログラムの形式的な特性が異なることが多く，単純な文字列検索や従来の類似コード検出技術では検出が困難である．そこで，本研究では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」と定義し，検出する手法を提案する．この手法は，ソースコードの形式的な特性に依存せず，プログラムの実行により実現される機能が類似しているコードの検出を目的とする．

2　機能的類似コード

機能的類似コードは，機能を構成する処理が類似するコード断片と定義する．このようなコードは，プロジェクト管理の不備や長期にわたるプロジェクトの進行に伴い，特に複数のチームや個人による独立した開発作業によって発生することがある．そのため，ソースコードの形式的な特性が必ずしも類似するとは限らず，開発者が機能的に類似したコードの存在に気づかない可能性が高い．実際に，独立した開発作業に起因して発生した類似コードが，構文的に類似している可能性が低いことを示す研究 [1] や，同じ機能を持つコードの検出に対して，構造の類似性だけでは不十分であると論じる研究 [2] もある．

3　提案手法

本研究では，コード片が呼び出すメソッドのセットの類似性に基づいて機能的類似コードを検出する手法を提案する．メソッドは特定の処理や操作を実施する一連の手続きを表現しており，これらを組み合わせて利用することで，特定の機能を実現する．したがって，利用するメソッドのセットが類似するコード片は，同様の機能を実現する可能性が高いと考えられる．特に外部ライブラリや API が提供するようなメソッドは，特定の機能やタスクを実現するために使用されることが多い．そこで，外部機能が提供するメソッドのセットに着目をし，その類似性に基づいて機能的類似コードの検出を行う．

提案手法の構成と処理の流れを 図 1 に示す．以降の各節では，各手順の詳細について 図 1 に基づいて述べる．

3.1　手順 1 ：メソッドの抽出

手順 1 では，各コード片を解析して呼び出すメソッドを抽出し，そのメソッドのセットを作成する．図 1 では，プロジェクト全体のソースコードのうち，上部が比較元コード片を表し，下部が比較対象となるコード片を表す．比較元コード片は開発者が任意のステートメントを選択し，比較対象となるコード片はソースコード中のすべてのブロックとする．比較元コード片 X ではメソッド x, y, z が抽出され，比較対象となるコード片 A ではメソッド x, q, コード片 B ではメソッド x, q, r が抽出される．これらのメソッドは各コード片が呼び出すメソッドのセットを形成する．

3.2　手順 2 ：呼び出し関係の解析

手順 2 では，手順 1 で抽出されたメソッドの呼び出し関係を解析し，外部機能が提供するメソッドを特定する．そして，メソッドのセットを外部機能が提供するメソッドを含むように置き換える．これは，呼び出し元のメソッドの機能はそのメソッドが呼び出すメソッドのセットによって表現できると考えたためである．この手順により，最終的に得られるメソッドのセットは，呼び出すメソッドが存在しないメソッドの集合となる．図 1 に示すメソッドの呼び出し関係では，比較元コード片 X が呼び出すメソッド z はメソッド p を呼び出すため，メソッドのセットを z から p に置き換える．同様にコード片 A が呼び出すメソッド q はメソッド y を呼び出すため，メソッドのセットを q から y に置き換える．

3.3　手順 3 ：類似度の算出

手順 3 では，手順 2 で置き換えられたメソッドのセットを利用して，各コード片が呼び出すメソッドのセットで共通するメソッドの割合を計算する．比較元コード片が呼び出すメソッドの集合 S_1 と比較対象となるコード片が呼び出すメソッドの集合 S_2 を比較した際の類似度を，(1) 式に示す定義により算出する．

$$Sim(S_1, S_2) = \frac{|S_1 \cap S_2|}{|S_1 \cup S_2|} \tag{1}$$

(1) 式の分母はメソッドの種類の総和を，分子は各セットで共通するメソッドの数を計算する．このような共通するメソッドの割合を用いた類似度計算によって，コード片が呼び出すメソッドのセットの類似性を表現できる．図 1 において，コード片 A との類似度は，メソッドの種類の総和

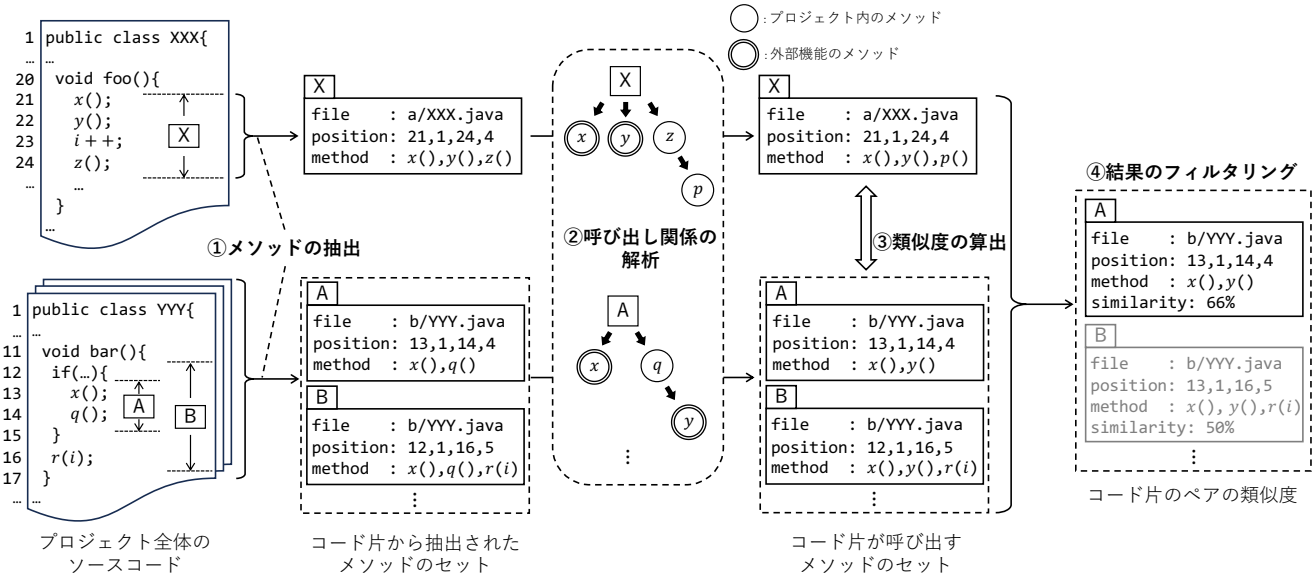

図 1　提案手法の構成と処理の流れ

表 1　robotium プロジェクトにおける Javadoc の類似度

グループ	70% 以上	70% 未満 40% 以上	40% 未満 10% 以上	10% 未満
N = 82,801	N = 6,104	N = 19,458	N = 17,118	N = 40,121
Javadoc の	37% ± 19%	28% ± 11%	26% ± 10%	25% ± 8%
類似度	30%[24%,46%]	26%[22%,32%]	25%[21%,29%]	24%[20%,28%]

*N は対象とした結果の数，基本統計量は平均値±標準偏差, 中央値 [第 1 四分位数, 第 3 四分位数] で表す.

が x, y, p の 3 種類，一致するメソッドの総和が x, y の 2 種類となるため，類似度は 2/3 (66%) となる.

3.4　手順 4 ：結果のフィルタリング

提案手法では，比較対象となるコード片が全てのブロックを対象としているため，同一のコード片を指す結果が複数存在する可能性がある．そこで手順 4 では，手順 3 で算出された類似度の結果のうち，重複する結果を除去する．具体的には 図 1 におけるコード片 A とコード片 B のように比較対象となるコード片同士がブロックの包含関係にある場合は，最も類似度の高い結果であるコード片 A の結果のみを残して他の結果を除去する.

4　評価実験

本研究では，機能的類似コードの検出可能性を評価するために，Java プログラムを対象として提案手法を実装した．そして，4 つの OSS プロジェクト[*1] を対象に，算出された類似度を 70% 以上，70% 未満 40% 以上，40% 未満 10% 以上，10% 未満のグループに分類し，各グループ間で Javadoc の類似度に有意差があるかを検証した．この際，コード片は Javadoc が取得可能なメソッドブロックのみを対象とした．そして，有意差検定にはノンパラメトリック検定の一種であるクラスカルウォリス検定を使用し，帰無仮説を「4 つのグループ間でメトリクス値に差がない」と設定した．その結果，いずれのプロジェクトにおいても p 値が 0.01 以下で帰無仮説を棄却し，各グループ間で有意差があることが確認された．例えば，RobotiumTech/robotium における各グループの Javadoc の類似度は 表 1 に示す結果となった．この結果から，類似度の高いグループの方が Javadoc の類似度が高く，Javadoc を利用した機能的類似性の判断では，提案手法によって機能的類似コードを検出できる可能性があると考えられる．また，類似度の高い結果を目視で観察したところ，実際に機能が類似しているケースが確認できた.

[*1] https://github.com/hub4j/github-api/releases/tag/github-api-1.318
https://github.com/88250/symphony/releases/tag/v3.6.4
https://github.com/datumbox/datumbox-framework/releases/tag/0.8.2
https://github.com/RobotiumTech/robotium/releases/tag/robotium-5.6.3

5　まとめ

本研究では，コード片が呼び出すメソッドのセットの類似性に基づいて機能的類似コードを検出する手法を提案した．そして，評価実験から提案手法が機能的類似コードを検出できる可能性があることが示唆された.

参考文献

[1] E. Juergens, et al. Code similarities beyond copy & paste. In *European Conference on Software Maintenance and Reengineering*, pp. 78 – 87, 2010.

[2] Y. Higo and S. Kusumoto. How should we measure functional sameness from program source code? an exploratory study on java methods. In *22nd ACM SIGSOFT International Symposium on Foundations of Software Engineering*, pp. 294 – 305, 2014.