2J-05

コードメトリクスを用いたソースコード評価に基づくプログラミング学習支援システム

伊藤 瑠哉† 大平 茂輝†† 長尾 確†††
名古屋大学 大学院情報科学研究科†
名古屋大学 情報基盤センター††
名古屋大学 大学院情報学研究科†††

1. はじめに

ソフトウェア開発において，品質の高いプログラムを開発することは重要である．プログラムの品質を把握するための手段として，ソースコードの状態を数値的に示すコードメトリクスが利用される．コードメトリクスを測定し，測定ツールや開発する環境が定めた閾値をもとに，クラスやメソッドのメトリクスの値の閾値からの超過を調べることで，品質が評価される．しかし，プログラム全体の品質を把握しやすくする仕組みが十分には検討されていない．

そこで本研究では，大規模なソースコード共有サービスである GitHub を対象に，リポジトリが持つ定性的・定量的な属性値とメトリクスの関係を分析し，各属性値とメトリクスの関係に基づいて，ソースコードを評価する機械学習モデルを作成した．

そして，作成した評価モデルを統合開発環境のプラグインとして組み込み，ソースコードを評価するシステムを開発した．

2. GitHub リポジトリを対象としたコードメトリクス分析

2.1. コードメトリクスの測定

本研究では，プログラミング言語 C#のソースコードを分析対象にした．ソースコードの解析には，.NET コンパイラープラットフォーム Roslyn を利用し，メソッド(コンストラクターやプロパティを除く)とクラスのコードメトリクスを測定するプログラムを作成した．本研究で測定するコードメトリクスを表 1 に示す．

表 1 測定するコードメトリクス

メソッド	クラス
コード行数	コード行数
Halstead Volume [1]	Halstead Volume
循環的複雑度 [2]	循環的複雑度
ローカル変数の数	フィールド数
	メソッド数

コード行数は論理コード行数のことを指す．クラスのコード行数，Halstead Volume，循環的複雑度は，関数の役割を持つメンバーの合計値とした．

2.2. データ収集

GitHub API v3 を利用して，C#のリポジトリに関する情報を収集した．API では以下の属性値が取得できる．

- Star 数
- Fork 数
- Issue 数(オープンになっている Issue 数)
- リポジトリ作成日
- リポジトリ更新日
- Branch 数
- Contributor 数
- Release 数

また，上記の値をもとに以下の情報を算出した．

- C#のリポジトリの作成数
- Commit 数
- リポジトリ更新期間(作成日から最終更新日まで)

Search API でプログラミング言語に C#を指定し，2017 年 6 月 23 日から 2017 年 7 月 17 日の期間で実施した結果，68 万個のリポジトリの情報が得られた．

収集したデータのうち，Star 数，Fork 数， Issue 数，Branch 数，Release 数，Commit 数が 1 以上のリポジトリに絞り込み，コードメトリクスの測定可能な 2946 個のリポジトリを分析対象にした．

Programming Learning Support System Based on Source Code Evaluation using Code Metrics
† Ryuya Ito ††Shigeki Ohira †††Katashi Nagao
†Graduate School of Information Science, Nagoya University
††Information Technology Center, Nagoya University
†††Graduate School of Informatics, Nagoya University

2.3. コードメトリクス分析

各リポジトリに対し，属性値をX軸，メトリクス値上位10%の中央値をY軸とし，プロットした場合，属性値が大きいリポジトリほどメトリクス値が小さく，属性値が小さいリポジトリの中にはメトリクス値が高いものが存在する傾向がみられた．リポジトリ作成日と更新日および更新期間以外の属性値で，同様の傾向がみられた．

3. ソースコード評価モデルの作成

3.1. 説明変数の作成

ソースコード全体のコードメトリクスの特徴を掴むために，メトリクス値のヒストグラムの割合を説明変数とした．

ヒストグラムの階級幅を決めるために，既存のメトリクスの閾値を 2 から 5 の間で分割した値を使用した．閾値以下の階級幅は，分割した値を使用し，閾値超過の階級幅は，分割した値の 0.5，1，2 倍の値を使用した．説明変数は 5 から 15 個に設定し，さまざまなバリエーションの説明変数を用意した．

最適な説明変数を決めるために，作成した評価モデルの MCC (Matthews Correlation Coefficient) の値を比較した．

3.2. 正解ラベルの付与

2.3 節で述べた傾向をもとに，リポジトリに対して，図 1 に示す分割の仕方で「良い」・「悪い」・「不明」の 3 つのラベルを付与した．

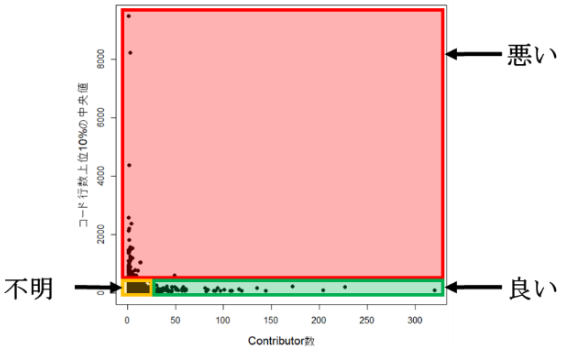

図 1 正解ラベルの付与のための分割方法

「良い」・「悪い」が付与されているリポジトリを機械学習に利用する．

3.3. 評価モデルの作成

本研究では，ロジスティック回帰分析を利用して，メトリクスごとに評価モデルを作成した．「良い」を 1，「悪い」を 0 として学習し，出力の確率値をコードの良さを表すスコアとして利用した．

4. プログラミング学習支援システム

本システムは，3 章で作成した評価モデルが組み込まれた Visual Studio のプラグインとして実装されている．

4.1. ソースコード評価の可視化

本システムは，現在のソースコードの評価が把握できるように，図 2 に示すレーダチャートを用いて，各メトリクスに対するスコア(青線)とスコアの最大値(赤線)を提示する．

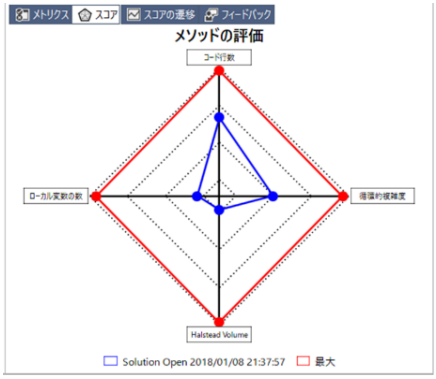

図 2 レーダチャートを用いた評価の可視化

また，ソースコードの改善度合いが把握できるように，折れ線グラフを用いて，スコアの遷移を提示する．

4.2. フィードバックによる改善支援

メトリクスに対するスコアに基づいて，フィードバックを行う(図 3)．スコアが悪い場合，メトリクス値が高いことによるデメリットを説明し，改善すべきメソッドやクラスを提示する．それらをクリックすると，該当箇所に遷移され，改善のためのアドバイスが提示される．

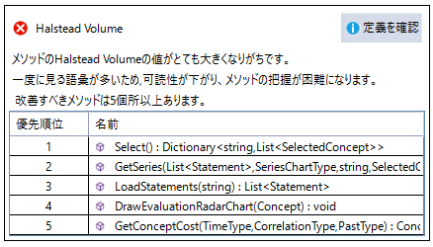

図 3 ユーザへのフィードバック

5 今後の課題

研究室の学生を対象に，システムの有無による，リファクタリング作業時におけるシステムの有効性を検証する予定である．

参考文献

[1] M. H. Halstead, “Elements of Software Science (Operating and Programming Systems series)”, Elsevier Science Inc., 1977.

[2] T. J. McCabe, “A Complexity Measure”, IEEE Transactions on Software Engineering, Vol. 2, No. 4, 1976.