7H-08

# コードメトリクスを用いた プログラミング能力の評価とその応用

伊藤 瑠哉† 竹島 亮† 長尾 確†

名古屋大学 大学院情報科学研究科†

## 1 はじめに

ソフトウェア開発において，作成したソースコードの品質評価を行うためにコードメトリクスが利用される．コードメトリクスの値の参考値として，オープンソースで計測した値を利用することが多い．

しかし，コードメトリクスの値とプログラミング能力を直接結びつけることは難しい．そこで，本研究ではオープンソースを対象にコードメトリクスを計測し，メソッドに対するコードメトリクスの値を 100 点満点で得点化を行う．

## 2 コードメトリクス

コードメトリクスとは，システム開発者が開発しているプログラムのソースコードを把握するためのソフトウェアの基準である．代表例として，コードの行数(LOC:Lines Of Code)，McCabe のサイクロマティック複雑度(Cyclomatic Complexity)[1]やオペランドやオペレータなどの語彙数に基づく Halstead メトリクス[2]などが挙げられる．

## 3 ソースコード解析

### 3.1 評価対象の言語

本研究での評価対象の言語は C#である．C#は Microsoft が開発したオブジェクト指向型のプログラミング言語で，.Net Framework 上で動作するアプリケーションの作成に利用される．

### 3.2 構文解析

本研究ではソースコード解析に Microsoft がオープンソースとして提供している.Net コンパイラープラットフォーム Roslyn[3]を利用する．Roslyn の標準ライブラリではコードメトリクスを計算するクラスがないため，構文木を元に自分で計算する必要がある．Roslyn によって，図 1 に示すようなコードは図 2 のような構文木として解析される．

```
Console.WriteLine("Hello Roslyn!!");
```

図 1 解析対象のコードの例

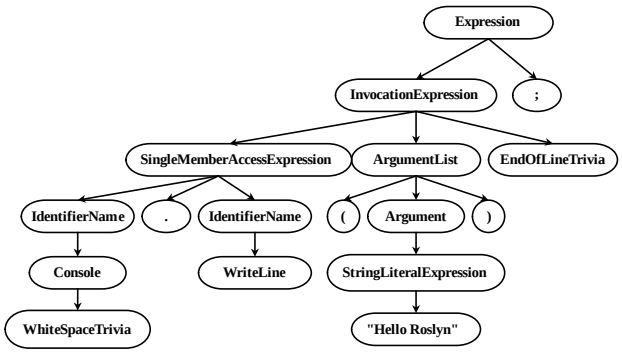


図 2 コードの構文木

### 3.3 解析対象のオープンソース

本研究では C#のライブラリとして代表的なオープンソース 3 種類(ASP.NET, Entity Framework, Newtonsoft.Json)を利用した．ASP.NET と Entity Framework は CodePlex から，Newtonsoft.Json は GitHub から入手できる．各オープンソースに関しての詳細を表 1 に示す．解析の際には，コンパイラによって生成されたコードや開発者が直接編集をしないコード，デバッグテストを行う際に利用するプロジェクトは除去した．

表 1 解析対象のオープンソースの詳細

| 名前 | ソリューション名 (プロジェクト数) |
|---|---|
| ASP.NET [4] | Runtime.sln (22) |
| Entity Framework [5] | EntityFramework.sln (6) |
| Newtonsoft.Json [6] | Newtonsoft.Json.sln (1) |

今回の解析で，3 つのプロジェクトで合計 29683 のメソッドを収集した．

## 4 ソースコードの評価

### 4.1 評価項目

本研究ではメソッドを対象に評価を行う．評価項目は以下の 4 点である

- コード行数
- サイクロマティック複雑度
- Halstead Volume
- 引数の数 (Number Of Parameters)

本研究で定義するコード行数は論理コード行数のことを指す．物理行数で計測した場合，作成者の記述スタイルに依存する可能性があるため，

Evaluation of programming skills based on code metrics
† ITO, Ryuya (rito@nagao.nuie.nagoya-u.ac.jp)
† TAKESHIMA, Ryo (takeshima@nagao.nuie.nagoya-u.ac.jp)
† NAGAO, Katashi (nagao@nuie.nagoya-u.ac.jp)
Graduate School of Information Science, Nagoya University (†)

論理コード行数を採用する．

### 4.2 ヒストグラムに基づく点数化

本研究では各評価項目に対して，ヒストグラムを作成し，点数化を行う．コード行数，サイクロマティック複雑度，引数の数に関しては階級幅 1 でヒストグラムをとる．コード行数に対してヒストグラムをとった場合，図 3 のようになり，頻度に偏りが生じてしまう(サイクロマティック複雑度，引数の数に関しても同様)．そのため，対数変換を用いて偏りを平滑化させる．

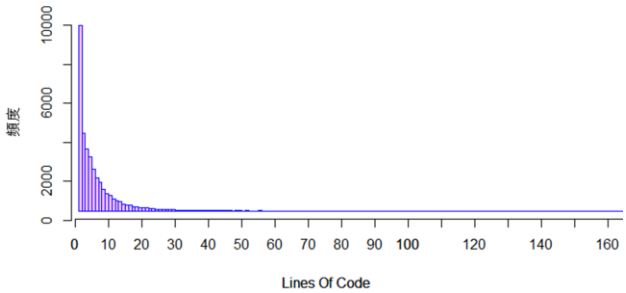


図 3 コード行数のヒストグラム

対数変換を行う際には，頻度が 0 の階級も存在するため，各階級の頻度に対して 1 を足す．そして，頻度を対数変換した値と階級から非線形回帰分析を行う．図 4 に対数変換した値と非線形回帰分析によって求まった多項式 (緑)を示す．

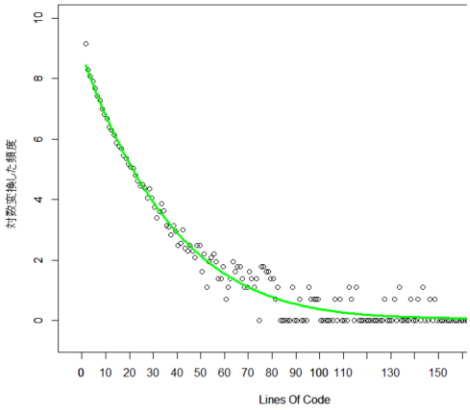


図 4 コード行数の頻度を対数変換した値と非線形回帰分析から得られる曲線

非線形回帰分析から得られる多項式をもとに最大値と最小値(この場合は 0)の差を求め，高さが 100 になるようにスケール変換を行う．これにより，100 点満点で点数付けを行うことができる．Halstead Volume の場合，算出される値が大きいため，あらかじめ対数変換を行い，ヒストグラムをとる．その結果を図 5 に示す．行数とは違い，頻度の偏りが生じていないため，頻度に対して対数変換を行わずに，非線形回帰分析を行う．その結果を図 6 に示す．100 点満点での点数付けは，行数と同様の手法をとる．

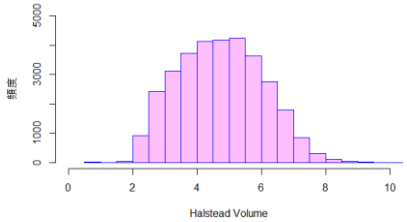


図 5 対数変換した Halstead Volume のヒストグラム

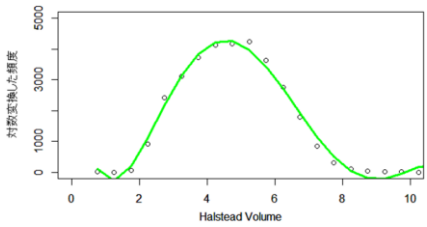


図 6 Halstead Volume の頻度を対数変換した値と非線形回帰分析から得られる曲線

従来のメトリクスの上限の参考値を，本手法で求まった計算式に当てはめると，表 2 のような結果になる．各参考値を考慮せずに記述するプログラマーは低い点数となり，ここから能力の評価ができると思われる．

表 2 メトリクスの上限の参考値に対する得点

| 名前 | 参考値(上限) | 得点 |
|---|---|---|
| コード行数 | 20 | 59 |
| サイクロマティック複雑度 | 20 | 39 |
| Halstead Volume | 1000 | 38 |
| 引数の数 | 6 | 57 |

## 5 おわりに

本研究ではメソッドに対しての評価手法を提案した．ソースコードにはプロパティ，フィールド，クラスといった様々な要素が存在するため，これらに対しても同様に評価が必要である．

そして，コードの評価機能を搭載したプラグインを開発し，評価の妥当性を検証するとともに，長期的にコードを評価・記録し，その遷移を分析する．

## 参考文献

1) T. J. McCabe, “A Complexity Measure”, IEEE Transactions On. Software. Engineering, Vol. 2, No. 4, 1976.
2) M. H. Halstead, “Elements of Software Science”, Elsevier Computer Science Library, 1977.
3) Roslyn, https://github.com/dotnet/roslyn
4) Microsoft ASP.NET, https://aspnetwebstack.codeplex.com/
5) Microsoft .NET Entity Framework, https://entityframework.codeplex.com/
6) Newtonsoft.Json, http://www.newtonsoft.com/json