5ZE-06

ソフトウェアメトリクスに基づくリファクタリング学習の支援

橋本 裕介† 酒井 三四郎‡
静岡大学大学院総合科学技術研究科† 静岡大学情報学部‡

1. 研究の背景・目的

プログラムの保守性に悪影響を与えかねない質の低いコードは「不吉な臭い」と呼ばれ、経験を積んだプログラマはその傾向を見つけてリファクタリングすることができる。しかし、経験の浅い人がそれを見つけることは難しく、無秩序なコードができやすい。ある程度規模の大きいプログラムを作るようになると保守性や可読性を考慮する必要が出てくるが、それまでとりあえず動けばいいプログラムを作ってきた人には難しく、学習が必要になる。

本研究では、ソフトウェアメトリクスを用いてリファクタリングすべき箇所を定量的に示し、その修正箇所に対してリファクタリング方法を提案することで、リファクタリングに不慣れな学習者を支援する。メトリクスに基づいたリファクタリングをさせることで、リファクタリングすべき箇所の特定とその修正を経験によって学習させる。学習によって自力で問題点を見つけ出し、適切にリファクタリングできるようになってもらうことを目的とする。

2. 関連研究

プログラムを定量的に評価し、改善すべき点を見つけるための指標としてソフトウェアメトリクスを用いる。本研究ではメソッド行数やサイクロマチック数などの基本的なメトリクスに加え、Chaidamber[1]らによる CK メトリクスを用いる。CK メトリクスはクラス間の関係を評価する 6 つのメトリクスから成る。これによって、サイクロマチック数のようなメソッド単位のメトリクスよりもプログラム全体の構造を見据えたコードの質を評価することができる。

Java で記述されたプログラムのメトリクスを計測するツールとして、Eclipse Metrics Plugin[2]があげられる。このツールは Eclipse のプラグインであり、選択したクラスやパッケージのメトリクスを計測して表形式で表示する。メトリクスの値が設定された閾値を超えると、そのメトリクスは赤色で強調表示される。

リファクタリングやメトリクスの見方に慣れた熟練者であれば、Eclipse Metrics Plugin のようなツールでメトリクスの値を見ながら問題点を見つけてリファクタリングすることができる。しかし、リファクタリング初心者にはメトリクスを見ただけではどこをどう修正すればいいのか、どこから手を付ければいいのかを判断するのは難しい。

表 1 使用するメトリクス

カテゴリ	名称	説明
CK メトリクス	WMC	クラスの複雑度
	NOC	子クラスの数
	DIT	継承ツリーの深さ
	LCOM	クラスの凝集度
	CBO	依存するクラスの数
	RFC	呼び出しうるメソッドの数
単純なメトリクス	メソッド行数	メソッドの行数
	サイクロマチック数	McCabe によるメソッドの複雑度
	ローカル変数の寿命	変数が使える範囲

3. リファクタリング学習支援ツールの提案

リファクタリングに不慣れな人に経験を積んで学習してもらうため、メトリクスの値を元にリファクタリングの方針を示すツールを作成した。ツールが提示する問題点と改善案を元にリファクタリングすることで、コードの質を判断する力と問題点をリファクタリングする力を身に付けさせる。本ツールはメトリクスに基づく問題点とその改善案をセットで提示するため、悪いメトリクスに対してどのように改善すべきかをリファクタリングしながら学ぶことができる。

作成中のプログラムに対してメトリクスを計測し、その値を元にリファクタリングすべき箇所と改善案を提案する。本ツールで計測するメトリクスは表 1 に示す 2 カテゴリ 9 種である。ローカル変数の寿命は、変数が宣言されてから変数が使えなくなるまでの行数を表す。

本ツールは Eclipse のプラグインとして動作し、エディタで編集中のファイルと同一パッケージのプログラムのメトリクスを計算する。計測したメトリクスの値を元に、リファクタリングすべき問題点と改善案をリファクタリング提案ビューに表示する。リファクタリング提案ビューの外観を図 1 に示す。

提示する問題点には問題の種類とメトリクスの値の大きさを元に重要度を設定し、重要な問題ほど上に表示される。例として、メソッド A の行数（30 行）とサイクロマチック数（17）が大きく、メソッド B は行数（32 行）が大きいというケースを考える。メソッドの行数を減らすには複数の小さなメソッドに分割することが多く、それに伴

図 1 リファクタリング提案ビューの外観

Learning support for refactoring based on software metrics
†Yusuke Hashimoto・Graduate School of Integrated Science and Technology, Shizuoka University
‡Sanshiro Sakai・Faculty of Informatics, Shizuoka University

って 1 メソッドあたりのサイクロマチック数も減少する。したがって、メソッド行数の削減をサイクロマチック数削減よりも優先すべきと考え、メソッド行数の削減の重要度が高いとする。また、メソッド A よりもメソッド B の行数がより重要な問題と考え、メソッド B の行数削減を最優先するべきだと考える。重要度の高い問題からリファクタリングしていくことにより、効率的に作業を進めることができる。修正が広範囲に及びやすい CK メトリクスを先にリファクタリングすることが望ましいため、CK メトリクスによる問題点は単純なメトリクスのものよりも上に表示する。

それぞれの問題点に対しては 1 つ以上の改善案が提示される。例えばメソッドのサイクロマチック数が大きいという問題には、役割ごとにメソッドを分割する、複雑な条件式を抽出してひとつのメソッドにする、などの改善案が提示される。CK メトリクスのひとつである WMC（クラス単位の複雑度）が大きいという問題には、サイクロマチック数と同様の改善案に加え、クラスを分割するという改善案が提示される。リファクタリング提案ビューの問題点をダブルクリックするとエディタ上で該当場所へジャンプすることができ、問題を把握してすぐにリファクタリングに取り掛かることができる。問題点は対応するメトリクスの値が閾値を超えた場合にリファクタリング提案ビューに表示されるため、値が十分良くなれば表示されなくなる。改善案をダブルクリックすると説明が表示され、具体的にどのような作業をすればよいか学ぶことができる。

本ツールはリファクタリングの支援ではなく学習の支援をするツールであるため、最終的には Eclipse Metrics Plugin のようなメトリクス計測ツールを使って適切にリファクタリングできるようになることを目的としている。

4. 評価実験

評価実験では既存のツール（Eclipse Metrics Plugin）と提案ツールを使ったリファクタリングをしてもらい、その学習効果を比較した。被験者は静岡大学情報学部の 4 年生 3 名で、全員に Java 及び Eclipse の使用経験がある。被験者にはツールの使い方、JUnit テスト、メトリクスについて説明し、メトリクスが評価指標のひとつであることを伝えた。まず、事前テストとして全員に既存のツールを使ってもらい、メトリクスを元にしたリファクタリングをしてもらった。次に、被験者を実験群と統制群に分け、実験群には提案ツールを、統制群には既存のツールを使ってリファクタリングしてもらった。最後に事後テストとして事前テストと同様に既存のツールでリファクタリングしてもらい、結果を比較する。事前テストと事後テストは 400 行程度のプログラムを 20 分で、比較実験では 500 行程度のプログラムを 30 分でリファクタリングしてもらった。プログラムには短時間で内容を把握するためにコメントで説明を書いてある。この実験はリファクタリングしてもらうのが目的であるため、敢えて冗長にしたり複雑にしたコードを含んでいる。このプログラムには JUnit によるテストコードが含まれており、実験終了時にテストが通るようにしてもらった。リファクタリングしてもらうプログラムの規模と被験者の能力を考慮し、この実験では複雑度（サイクロマチック数と WMC）とメソッドの行数を中心に評価する。

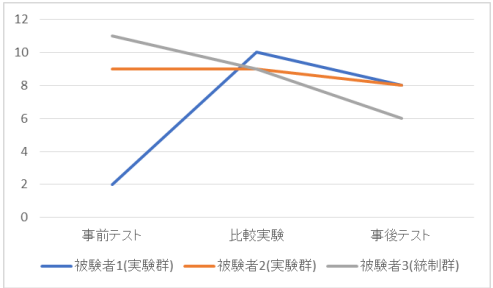

図 2　リファクタリングによる WMC の減少数

5. 結果と考察

被験者がリファクタリングしたプログラムのメトリクスを計測した。評価指標としてリファクタリングの前後でどれだけ WMC が減少したかを比較する。事前テスト、比較実験、事後テストにおいて、WMC の合計がどれだけ減少したかを図 2 に示す。事前テストでは統制群が最も良い結果を残したのに対し、事後テストでは実験群が両者とも統制群以上に WMC を減少させた。実験群は提案ツールを使用した後に WMC をより大きく減らせるようになっている事がわかる。特に事前テストで良い結果の出なかった人の変化が大きい。比較実験で提案ツールを使用した後の改善度合いが統制群よりも大きいため、提案ツールによってリファクタリングの方法を学び、提案ツールを使わない事後テストでも良い結果を出すことができたといえる。メソッド行数の変化については、被験者によって増えたり大きく減らしたりと違いが見られたが、使ったツールや提案ツールの使用前後による違いは見られなかった。

実験後のアンケートでは、提案ツールがリファクタリングに役立つかという質問に対して、実験群 2 名全員が「そう思う」と答えたのに対し、リファクタリングの学習に役立つかという質問には 1 名が「とてもそう思う」、もう 1 名が「どちらとも言えない」と答えた。リファクタリング自体の支援ではなく学習の支援をするためのツールとしては課題の残る結果といえる。

既存のツールと比較して優れている点としては「改善の方針が示されるので作業しやすいと感じた」という回答が得られた。また、ツールを使用した感想として「自分のコードを可読性や保守性高く実装するのに大変有効だと感じた」という回答も見られた。リファクタリングの方針を示して支援するという点では期待したとおりに使われているといえる。しかし、UI の使いづらさを指摘する意見もあり、ツールとして改善すべき点も残っている。

6. まとめ

本研究では、計測したメトリクスに基づいてリファクタリングの方針を示すことで、リファクタリング初心者の学習を支援するツールを作成した。実験結果からはリファクタリングの手法を学ぶのに一定の効果があると解決できるデータが得られた。

今後は引き続き評価実験を行い、実験結果の精度を高める必要がある。

参考文献

[1] Chidamber, Shyam. R., Kemerer, Chris F. : A metrics suite for object oriented design, IEEE Transactions on software engineering, 20(6), pp.476-493 (1994)

[2] Metrics 1.3.6: http://metrics.sourceforge.net/, 2018 年 1 月 10 日アクセス