4L-3

類似度メトリクスを用いたリファクタリング支援ツールの試作

木川田 惇† 杉山 安洋‡

日本大学大学院工学研究科† 日本大学工学部情報工学科‡

1. はじめに

ソフトウェア開発においてリファクタリング[3]はかかせない技術である．特にシステムが大規模なものになると，ソースコードの量も膨大になるため，リファクタリングを行い，保守しやすい構造に改善する必要が発生することが多い．一般的によく行われるリファクタリングの手法の一つに，重複コードや類似コードの排除がある．同じような処理部分を個別に記述するよりも，一箇所にまとめた方が，将来的な修正を考えた場合効果的である．

ここまで我々の研究室では，Java ソースコード間の類似度を求める際に，自己組織化マップ[2]を用いてソースコード群を分類し，その分類をもとにソースコード間の類似度を求める機能を持ったライブラリ[1]と，それを用いたツール CodeMap を開発してきた．現在，リファクタリングの支援のために，CodeMap を用いて類似コードを分類し，類似コードのマージを提案するツールを試作している．本論文は，そのツールの概要を述べるものである．

2. CodeMap の概要

CodeMap は Java ソースコード間の類似度を測定するツールである．具体的には，ソースコードからメトリクスを抽出し，その値を自己組織化マップと呼ばれるニューラルネットにより分類し，結果を二次元の図で表示する．測定は，クラス単位やメソッド単位の他，for 文や while 文等の Java の構文要素単位で行うことが可能である．また，測定に用いるメトリクスは，ユーザが自由に定義できるようになっているため，様々なメトリクスを用いた測定が可能である．

図 1 は CodeMap 自身の Java のソースコードの類似度をクラス単位で測定したものである．図中の数字はソースコードにつけられた番号である．類似と判定されたソースコードは図中では近い位置に表示される．隣接するソースコードの類似度が高いほど図中では白く表示される．

図 1.CodeMap の実行例

3. リファクタリング支援ツールの概要

CodeMap を用いたリファクタリング支援ツールとして，ソースコード中の類似部分を抽出し，そのマージをユーザに提案するツールを試作した．今回は，類似メソッドが異なるクラスに存在する場合，それらのメソッドのマージと，それをもとにした新しいクラスの作成を提案する機能を実装した．それぞれの類似メソッドをマージしたメソッドを持つクラスを作成すれば，もととなるクラスと継承・集約関係を持たせることにより，類似メソッドを個別に記述する必要がなくなる．

3.1. 提案機能の概要

図 2 は CodeMap 自身のソースコードを本ツールにかけ，提案された結果の画面である．それぞれの四角がクラスを表し，その中に記されているものがそのクラスが持つメソッドである．上段が新しく作成することを提案しているクラスで，下段が提案もととなるクラスである．下段のクラスの黒色以外のメソッドの色が同じものが類似していることを表している．これは，GrainFileParser と KeywordFileParser というクラスの readNameValue というメソッド同士と，readLabelValue というメソッド同士が類似しているので，それぞれのメソッド同士のマージと，それらを含む新しいクラスを作成することを提案している．このような提案を類似度測定の結果から複数提示する．ユーザは，画面下の prev ボタン，next ボタンによって，画面を切り替えて複数の提案を確認することが出来る．

3.2. 提案の作成手法

本ツールでは，類似したメソッドをもとに，

Implementation of a Refactoring Supporting Tool Using Similarity Metrics.
†Jun Kikawada, Graduate School of Engineering, Nihon University
‡Yasuhiro Sugiyama, Department of Computer Science, College of Engineering, Nihon University

新しいクラスの作成を提案する．類似結果からは，数多くの提案クラスが考えられる．本来であれば，ユーザにとって有効なクラスのみを提案することが望ましい．しかし，有効性の判定は現状では困難である．そこで今回は，測定対象のクラスの組み合わせを総当りで考え，全ての提案結果を提示することにした．

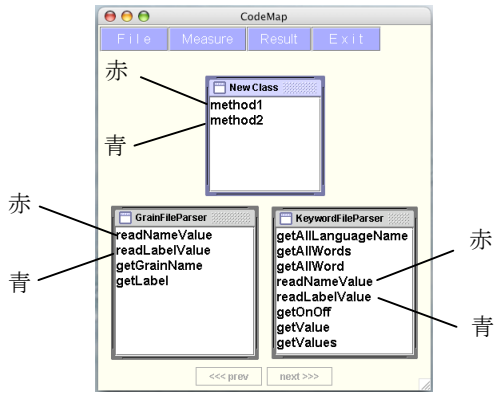

図 2.クラスの提案

提案を作成する際に，本ツールは以下のような処理を行っている．まず，対象となるシステム中のすべてのメソッドの類似度を CodeMap により測定し，類似メソッドを抽出する．続いて，測定対象となるシステム中のクラスの複数個の組み合わせに対して，それらに共通する類似メソッドがあるかどうかを判定し，もし存在する場合には，それらのメソッドを持つクラスを作成することを提案する．

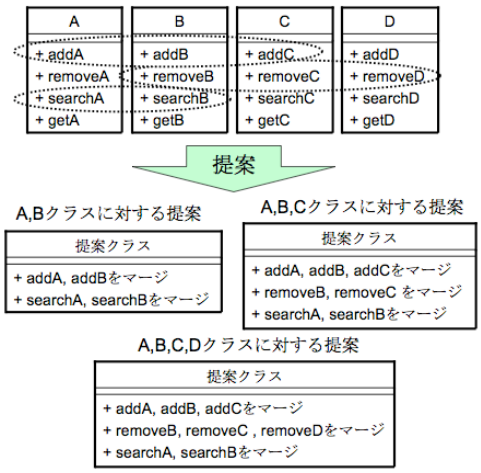

図 3.提案クラスの例

例えば，図３は４つのクラスから構成されるシステムを本ツールで解析している場合の提案結果の一部である．図中では，点線で囲んだ addA，addB，addC 等のメソッドは類似と判定されたメソッドである．本例では，４つのクラスを対象としているため，それらのクラスの複数個の組み合わせ AB，AC，…，ABC，ABD，…，ABCD に対して，類似メソッドがあるかどうかを判定し，存在する場合には，それらを含むクラスを作成することを提案している．なお，提案するクラスに持たせるメソッドの数は，該当するクラス間に存在する全ての類似メソッドをマージしたものを含み，メソッドのどれかを省略したいという場合の判断は提案結果を見たユーザに任せることにした．

4. 適用実験

今回は，行数，予約語の数，演算子の数，局所変数の数，メソッド呼び出しの数という基本的なメトリクスを用いて，提案の作成実験を行った．前述した図２がその結果の一部である．類似度の分類結果から手作業で提案を作成した結果と比較し，当初の計画通りの提案がされていることを確認できた．

さらに，提案された結果をもとに，実際にマージしたメソッドを持つクラスを作成した．もととなるクラスから類似メソッドの記述を排除し，作成したクラスをスーパークラスとすることで，設計を改善することができた．なお，新しいクラスを作成する際に，メソッド中で参照されているインスタンス変数を新しいクラスに持たせる必要があったため，それぞれのクラスのインスタンス変数を新しいクラスに持たせ，private だった修飾子を protected に変更する必要が発生した．

5. 今後の課題

今回の適用実験で，提案結果の中には，実際にマージできないものや設計上有効でない提案が含まれていた．特に問題となるのは，インスタンス変数の扱いである．類似メソッドをマージする際には，それらが参照しているインスタンス変数もマージできるようにする必要があるが，現在ではその作業はユーザにまかせている．今後は，メソッドとインスタンス変数の関係を含めて，提案を行えるようにツールを改善していく予定である．

参考文献

［１］ 北川俊広，杉山安洋，自己組織化マップを用いたJavaソースコード間類似度測定ライブラリの試作，信学技報，電子情報通信学会，2005.

［２］ Kohonen, T., Self-Organizing Maps (second edition), Series in Information Sciences, Vol.30, Springer-Verlag,1997.

［３］ Martin Fowler et. al, リファクタリング：プログラムの体質改善テクニック，ピアソン・エデュケーション，2000