5L-2

フィールド・メソッド関係による javaプログラムのリファクタリング支援

吉田　信一朗[†1]，　紫合　治[†2]
東京電機大学大学院情報環境学研究科[†1]，東京電機大学情報環境学部[†2]

1．はじめに

現在，ソフトウェアメトリクスやリファクタリングツールのようなソフトウェア開発後にソフトウェアの尺度を計っていきソフトウェアやプログラムの品質を上げていくものが研究されている．[1]

ここでは，リファクタリングとして活用できるよう，表や島づくりを通してフィールド・メソッド間の関連から解析をしていく方式とツールについて報告する．

2．提案手法

2.1　表作成の概要

フィールド視点からのメソッド毎のフィールドの動きに関する表にする．この表をFM表と呼ぶ．従来のソースコードを読み解析していく方式はメソッド視点からプログラムの作成を行っていたが，本研究ではフィールド視点からの設計・デバッグを可能にする表を提案する．従来のメソッド視点の考え方の表を表1に，フィールド視点の考え方の表を表2に示す．

表1は，従来のエディタ等でプログラムする際にメソッド毎にプログラムをしていく考え方を表したものである．

表2は，従来の考え方とは違い，フィールド毎にメソッドでどのような変更・参照をされているのかを表したものである．これは変数のアクセスに着目し解析しているところがプログラムスライシング[2][3]に似た考え方となる．

表.1 メソッド視点での考え方

	id	name	count	cflag
初期値			count=0;	cflag = 0;
C1()		name = “”;		
setID()	this.id = id;		count++;	cflag = 1;
getID()	return id;			

表.2 フィールド視点での考え方

	id	name	count	cflag
初期値			count=0;	cflag = 0;
C1()		name = “”;		
setID()	this.id = id;		count++;	cflag = 1;
getID()	return id;			

2.2　島づくり

クラス分けが十分になされておらずクラスの持つ意味が明確になっていないプログラムは，良いプログラムとはいえない．より分かりやすいプログラムを作成するために，クラス分けの候補を提示することができるように，このフィールド・メソッド間の関連を利用できないか考えた．フィールドとメソッドの関係を重み付け，関係の強いものを1つのクラス候補として分けるためにグラフ理論の極点集合[4]を応用しプログラムを作成した．

図1に極点集合を用いた島づくりの方法例を示す．

・頂点：メソッド・フィールド
・辺：メソッド・フィールド間の関係
　　メソッド呼び出し
・辺の重み：フィールドとメソッドの関係の強さ
　　フィールドの代入を 12
　　フィールドの参照を 10
　　フィールドのメソッド呼び出しを 11
　　ローカルメソッド呼び出しを 4

としてグラフを作成する．

頂点と辺と重みで作成されたグラフに最小カット・極点集合を応用した例を図1に示す．点線で囲まれた4つの大きな円がクラス分割の候補として提案される．この時のクラス分割のアルゴリズムとして，点線の中のカット数が点線のカット数より大きくなるようにクラス分割されている．カット数とは，頂点または頂点の集合から出る辺の重みの集合である．

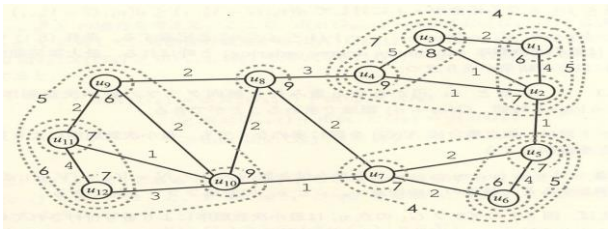

図.1島づくりの方法

3．開発

提案手法を用いて，プログラム作成後にソースコードから表・島を自動生成するツールの開発を行った．

3.1　ツールの構成

ツールの構成を図2に示す．

ソースコードの解析を行うにあたり，eclipse の JDT の ASTParser を用いた．ASTParser は java の構文解析を行いフィールドやメソッドの宣言，メソッドの中身等を正確に読み取れる．

まず最初に java のソースコードを読み込み，ASTParser による構文解析を行う．

次に構文解析からメソッド宣言・フィールド宣言の情報からフィールド名とメソッド名を抽出する．

その後，メソッドの中身を解析していく．フィールドアクセスがあった場合に代入なのか参照なのか，またはフィールドのメソッド呼び出しなのかを判別して記録する．

表の生成では，解析したフィールド名・メソッド名とその関係を代入，参照を分かりやすく表にして表示している．

クラス分割では，解析データから極点集合の応用を

Refactoring Support for Java Program by Fields and Methods Relations
†1 Shinichiro Yoshida　and　†2 Osamu Shigo
†1 Graduate school of Information Environment , Tokyo Denki University.
†2 School of Information Environment , Tokyo Denki University.

用い，繋がりの強いフィールド・メソッドのグループ分けを行ないクラス候補の提示をする．

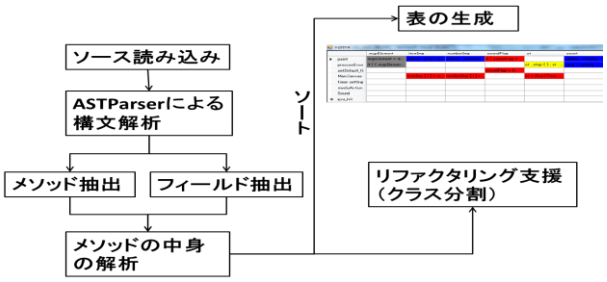

図.2　ツールの構成

3.2　FM表の詳細

本ツールで表示される表の例を図3に示す．

ここで表示される色については参照・代入・呼び出しの組み合わせで以下の8種類がある．

(1) 赤色:フィールドの変更
(2) 青色:フィールドの参照
(3) 黄色:フィールドのメソッド呼び出し
(4) 紺色:ローカルメソッドの呼び出し
(5) 水色:(1)＋(3)
(6) 灰色:(1)＋(2)
(7) 桃色:(2)＋(3)
(8)緑色:(1)＋(2)＋(3)

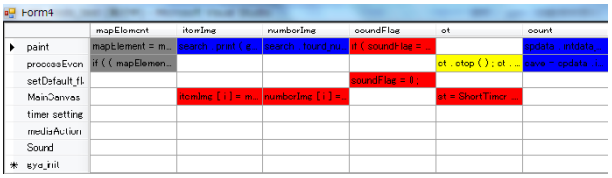

図3.　本ツール使用時の表

3.3　島づくりの詳細

リファクタリング支援のためのクラス分割支援として，フィールドとメソッドの島分けがある．

参照，代入，メソッド呼び出しの関係にそれぞれ重みを付けて重みの強いフィールド・メソッド同士を2.2で述べた方法を用いてクラス分けの指標としていく．

島づくりの例を図4に示す．「r」の後に続く数字は他の島との関わりの重さを表しており，「{」の後に島分け候補となるフィールド。メソッドを表す．この例では，一番大きく括った島分け候補として，フィールド「y1,y2,y3」とメソッド「setY,getY」の島と，フィールド「x1,x2,x3,z」とメソッド「setX,getX,plus」の島の2つの島分けが候補となる．

```
r:0.0 {f-y1 f-y2 f-y3 m-setY m-getY
    r:22.0 {f-y1
    r:22.0 {f-y2 f-y3 m-setY m-getY
        r:22.0 {f-y2
        r:22.0 {f-y3
        r:36.0 {m-setY
        r:30.0 {m-getY
r:0.0 {f-x1 f-x2 f-x3 m-setX m-getX f-z m-plus
    r:10.0 {f-x1 f-x2 f-x3 m-setX m-getX
        r:32.0 {f-x1
        r:22.0 {f-x2 f-x3 m-setX m-getX
            r:22.0 {f-x2
            r:22.0 {f-x3
            r:36.0 {m-setX
            r:30.0 {m-getX
    r:10.0 {f-z m-plus
        r:10.0 {f-z
        r:20.0 {m-plus
```

図. 4島づくりの例

4．評価

4.1　FM表の解析内容

図3のFM表を解析した際の解析例を挙げる．

(1)フィールドと全く関わりがなく他のメソッドとも関係のないメソッドに関しては，他のクラスにあっても問題のないメソッドの可能性が高いため，他のクラスへの移動候補になる．

(2)スタティックフィールドとしか関わっていないメソッドはスタティックなメソッドにすべきである．

(3)多くのフィールドを参照・代入しているメソッドは，複雑すぎる可能性があるのでメソッド分けをするべきである．

(4)1つのフィールドとのみ関係するパブリックなメソッドはセッター・ゲッターの可能性が有る．その場合には他のメソッドから直接フィールドをアクセスするべきではない．

4.2　FM表の評価

FM表の評価として，筆者が以前作成したプロジェクトを対象に行った．[5]

その結果では，熟練度により「フィールドのメソッド出現率」と「メソッドのフィールド使用率」に特徴が出た．

「フィールドのメソッド出現率」は，1つのフィールドが何％のメソッドで出現しているかを表した指標であり，「メソッドのフィールド使用率」は，1つのメソッドが何％のフィールドを使用しているかを表した指標である．

4.3　島づくりの評価

オープンソースを対象とした島づくりの評価を行なったが，期待していたような島はできなかった．

対象プロジェクトが複雑な作りになっておらず綺麗なクラスに既になっている，または関係の重み付けが適正ではないということが原因として考えられる．

5．まとめ

本研究では ASTParser による java のプログラム解析をFM表を用いて行ってきた．極点集合による島づくりでは良い結果が得られなかったため，重み付けの最適化等の強化をしていく必要がある．

参考文献

[1] 石川洋，“リファクタリングによるソフトウェアの品質向上に関する考察”福山大学人間文化学部紀要 8, 1-9, 2008-03.

[2] M.Weiser, “Program slicing”, Proc. 5th International Conference on Software Engineering, pp.439-449, 1981.

[3] 高田智則　井上克郎　大畑文明　芦田佳行，“制限された動的情報を用いたプログラムスライシング手法の提案”，電子情報通信学会論文誌 D-1 Vol.J85-D-1 No.2 pp.228-235, 2002.

[4] 茨木俊秀　永持仁　石井利昌，“グラフ理論―連結構造とその応用―”，朝倉書店

[5] 吉田信一朗　紫合治，“フィールド・メソッド関係表によるプログラムの評価”，情報処理学会第 179 回ソフトウェア工学研究発表会．