リファクタリングのソフトウェア保守に対する効果測定

松本　義弘†　肥後　芳樹†　Livieri Simone†　片岡　欣夫†　楠本　真二†　井上　克郎†

大阪大学大学院情報科学研究科†

1．はじめに

近年，ソフトウェア保守作業の効率化を目的に，リファクタリング[1]と呼ばれる技術が注目されている．リファクタリングとは，ソフトウェアの外部的振る舞いを保ったまま内部の構造を改善する技術である．本稿では，リファクタリングがソフトウェア保守に与える効果を計測する手法を提案する．

2．現状の問題

リファクタリング技術に関する研究は数多く行われているが，その多くはリファクタリング箇所の特定やソースコードの自動修正に関するものであり，リファクタリング効果の測定に関する研究はあまり行われていない．開発者や保守管理者がこれまでの経験などに基づいて主観的に評価を行っているのが現状である．

また，リファクタリングを適用することにより，ソフトウェアのさまざまな特性が変化するため，それらを総合的に評価することが必要である．例えば，メソッドの一部を新たなメソッドとして抽出した場合（「メソッドの抽出」パターンの適用），メソッドが小さくなるために複雑度は小さくなるが（改善する）が，メソッドの数が多くなるため，凝集度が小さくなる（悪化する）ことが考えられる．ソフトウェアの特性を定量的に表す指標としてこれまでに多くのメトリクスが提案されているが，各特性を表すメトリクスは単位が異なるために単純に比較を行うことが難しい．本稿では，メトリクスの偏差値を用いてリファクタリング効果を総合的に測定する手法を提案する．

3．提案手法

本節では，提案手法について述べる．3.1節では，提案手法で用いる CK メトリクス[2]の説明を行い，3.2節では測定方法について述べる．

3.1 CK メトリクス

CK メトリクスは，6 種のメトリクスから構成される．本節では，メトリクスを取得するツールの実装上，取得できたメトリクス WMC，CBO，LCOM*を説明する．これらのメトリクスは，いずれも測定値が大きいほど，ソフトウェア品質が劣っていることを表す．

3.1.1 WMC（クラスの重み付きメソッド数）

あるクラス C_i が，メソッド $M_1,\cdots,M_n$ を持つとする．これらのメソッドの複雑度をそれぞれ $c_1,\cdots,c_n$ とする．このとき，$WMC=\sum c_i$ である．メソッドの複雑度には，Halsted のメトリクス，McCabe のサイクロマチック数[3]などが用いられる．

3.1.2 CBO（クラス間の結合）

計測対象クラスが結合しているクラスの数である．あるクラスが他のクラスのメソッドやインスタンス変数を参照しているとき，結合しているという．

3.1.3 LCOM*（クラスの凝集の欠如）[4]

あるクラスの凝集の欠如を表す．定義は次の式の通りである．

$$LCOM^*=\frac{\left(\frac{1}{a}\sum_{j=1}^{a}\mu(A_j)\right)-m}{1-m}$$

a と m はそれぞれ対象のクラスに定義されているフィールドの数とメソッドの数である．A_j は j 番目のフィールド，$\mu(A_j)$ はフィールド A_j を用いているメソッドの数を表す．

3.2 アプローチ

CK メトリクス(WMC, CBO, LCOM*)を用いて，ソフトウェア保守に与える効果を測定する手法を述べる．本手法は以下の 5 つの STEP からなる．

STEP1：対象プログラムの全クラスの WMC, CBO, LCOM*を計算し，メトリクスごとに平均，標準偏差を計算する．

STEP2：リファクタリング対象のクラスに対し，各メトリクスの偏差値を求める．

STEP3：リファクタリングを適用し，再び対象クラスの各メトリクスの偏差値を計算する．

STEP4：STEP2, 3 の結果から，リファクタリング対象の全クラスの *Eval* を計算する．

STEP5：STEP4 の結果から，*TEval* を求め，総合的にリファクタリングの効果を評価する．

次に，*Eval* と *TEval* を定義する．

$$Eval(A)=\sum_{i\in M}(div_i'(C_i'(A))-div_i(C_i(A)))$$

where,

$M=\{WMC, CBO, LCOM^*\}$

Measuring the Effectiveness of Refactoring for Software Maintenance

†Yoshihiro MATSUMOTO, †Yoshiki HIGO, †Livieri Simone, †Yoshio KATAOKA, †Shinji KUSUMOTO, †Katsuro INOUE

†Graduate School of Information Science and Technology, Osaka University

$C_i(A)$ ： クラスAのメトリクスiの実測値

$div_i(J)$：全クラスのiに対する実測値Jの偏差値

（$C_i(A)$, $div_i(J)$はリファクタリング前の値，$C_i'(A)$, $div_i'(J)$はリファクタリング後の値）

リファクタリングの対象となるクラス群が$c_1,\cdots,c_n$のとき，$TEval$を次のように定義する．

$$TEval = \sum Eval(c_i)$$

最後に，$TEval$が表すリファクタリングの効果を次のように定義する．

$TEval < 0 : good$

$TEval = 0 : no\ change$

$TEval > 0 : bad$

4．適用実験

jakarta-tomcat-5.5.4（クラス数:1310）において，検出されたBad-Smell（1つの重複コード）に対し，リファクタリングパターンの1つである「メソッドの引き上げ(Pull Up Method)」を適用する．「メソッドの引き上げ」とは，同じ結果をもたらすメソッドが複数のサブクラスに存在した場合，それらを親クラスに引き上げるリファクタリングである．最も単純なケースは，複数のメソッド本体が全く同じである場合である(図1参照)．重複したコードが兄弟クラスに存在した場合には，メソッドの抽出を行ってから，メソッドの引き上げを行えばよい．また，メトリクスはEclipseのプラグインである"Eclipse Metrics Plugin"[5]を利用して取得する．

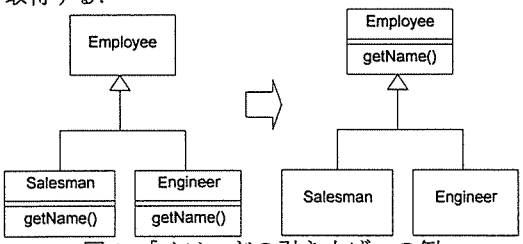

図1.「メソッドの引き上げ」の例

本実験では，リファクタリング対象のクラスをA,B,Cとし，クラスB,Cに重複するメソッドをクラスAに引き上げる．リファクタリングの適用前後におけるメトリクスの実測値および偏差値を求め，表1に示す．

表1よりクラスA,B,Cに対して，$Eval$を算出する．その結果を表2にまとめる．

表2．リファクタリング前後の偏差値の変化

	CBO	LCOM*	WMC	*Eval*
A	+3.08	0.00	+1.05	+4.13
B	−2.05	0.00	−1.04	−3.09
C	−2.05	0.00	−1.04	−3.09
TEval				−2.05

「メソッドの引き上げ」の適用によって，メソッドが引き上げられたクラスB,Cの$Eval$は共に-3.09，メソッドが追加されたクラスAの$Eval$は+4.13である．この結果から$TEval$を求めると，-2.05という結果が得られた．つまり本手法では，このリファクタリングを適用することで，ソフトウェア保守の難しさを改善したといえる．

5．まとめと今後の課題

本稿では，リファクタリングがソフトウェア保守に与える効果を測定する手法を提案した．今後の課題は，CKメトリクスで今回使用できなかったメトリクスを用い，他のリファクタリングに対しても本手法を適用することである．また，本手法の妥当性を検証する必要があると考えている．

参考文献

[1] M. Fowler, Refactoring: improving the design of existing code, Addison Wesley, 1999.

[2] Chidamber, S. and Kemerer, C.: A Metrics Suite for Object Oriented Design, IEEE Transaction on Software Engineering, vol.20, No.6 (1994)

[3] 山田茂，高橋宗雄：ソフトウェアマネジメントモデル入門－ソフトウェアの品質の可視化と評価法，共立出版株式会社(1993)

[4] Henderson-Sellers, B., Object-oriented metrics : measures of complexity, Prentice-Hall, pp.142-147, 1996.

[5] Eclipse Metrics Plugin. http://www.teaminabox.co.uk/.

表1．リファクタリング前後における各メトリクスの実測値および偏差値

		CBO			LCOM*(%)			WMC		
		前	後	差分	前	後	差分	前	後	差分
A	実測値	36	39	+3	95	95	0	108	113	+5
	偏差値	75.86	78.94	+3.08	62.91	62.91	0.00	67.24	68.29	+1.05
B	実測値	30	28	−2	91	91	0	137	132	−5
	偏差値	69.71	67.66	−2.05	61.96	61.96	0.00	73.30	72.26	−1.04
C	実測値	39	37	−2	91	91	0	108	103	−5
	偏差値	78.94	76.89	−2.05	61.96	61.96	0.00	67.24	66.20	−1.04