動的情報を利用したソフトウェア部品評価手法

藤井　将人[†]　　横森　励士[†]（学生員）
山本　哲男[††]　　井上　克郎[†††]（正員）

Software Component Ranking using Dynamical Relation

Masato FUJII[†], *Nonmember*,
Reishi YOKOMORI[†], *Student Member*,
Tetsuo YAMAMOTO[††], *Nonmember*, and
Katsuro INOUE[†††], *Member*

[†] 大阪大学大学院基礎工学研究科, 豊中市
Graduate School of Engineering Science, Osaka University
1–3, Machikaneyama-cho, Toyonaka, Osaka 560–8531, Japan
[††] 科学技術振興事業団, 川口市
Japan Science and Technology Corporation
4–1–8, Honmachi, Kawaguchi-shi, Saitama 332–8531, Japan
[†††] 大阪大学大学院情報科学研究科, 豊中市
Graduate School of Information Science and Technology, Osaka University
1–3, Machikaneyama-cho, Toyonaka, Osaka 560–8531, Japan

あらまし　本論文では，動的な情報を利用したソフトウェア部品評価手法を提案する．本手法を用いることで，ソフトウェア実行毎に利用された部品の重要度を計算することができ，特定機能を実装している部品の理解・再利用を支援することが可能となる．

キーワード　再利用, 動的支配関係,Component Rank 法

1. まえがき

ソフトウェア開発において既存のソフトウェア部品を再利用することで，生産性と品質を改善し，結果として開発コストを削減することができる．我々は，開発者に部品選択のための基準を示す部品評価手法として，静的な部品間の利用実績に基づいたソフトウェア部品評価手法 (Component Rank 法, CR 法) を提案し，その有効性を確認している [1], [2]．しかし，各部品が「いつ」「どのように」利用されているかという情報がなければ，実利用方法がわからないユーザにとって，再利用の恩恵を受けることができない．

そこで本論文では，動的な情報を利用した部品評価手法の提案を行う．ソフトウェア実行時に得られる動的支配関係を用いることで，単に各部品間の呼び出し関係の推移も得られるだけでなく，部品の支配関係を定量化した値が得られ，対象システムの振る舞いを理解するのに有効な手法であると考えられる．この提案手法を基に，Java アプリケーションを対象として，部品評価値，支配関係を求めるシステムを実現し，有効性を確認する．また，求めた部品評価値でフィルタリングを行い，重要な部品間のみのシーケンス図を作成するツールを実現する．注目すべき部品を絞り込んで開発者に表示することで，ソフトウェア理解・再利用を支援することが可能となる．

2. 動的情報を利用したソフトウェア部品評価手法

本手法による部品評価値計算手順を以下に示し，以降，各計算手順について説明を行う．

（1） ソフトウェアを実行し，動的支配関係を取得
（2） 動的支配関係から部品グラフを生成
（3） CR 法を用いて部品評価値計算

2.1 動的支配関係

一般に**ソフトウェア部品**(Software Comopnent) とは，再利用できるように設計された部品とされている [3]．本論文ではより一般的に，開発者が再利用を行う単位をソフトウェア部品，あるいは単に部品と呼ぶ．

部品間には実行時，互いに呼び出す，呼び出されるという 2 項関係が存在する．この関係のことを**動的支配関係**と呼ぶ．入力データに依存して実行経路も異なるため，それに伴い動的支配関係も変化する．この動的支配関係は，実行履歴を保存することで取得できる．メソッド呼び出しや例外等，指定した呼び出しイベントが発生した際，呼び出した部品と呼び出された部品の対を支配関係として抽出する．

2.2 動的部品グラフ

ソフトウェア部品を頂点，動的支配関係を呼び出される部品から呼び出した部品への有向辺としたものを，**動的部品グラフ**(*Component Graph*) と呼ぶ． 図 1 にその例を示す．

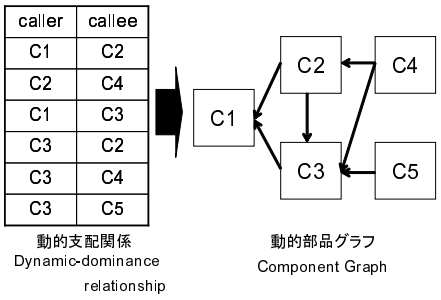

図 1　動的部品グラフ例
Fig. 1　Example: Dynamic Component Graph

2.3 部品評価値計算法

ここでは，部品グラフを入力とし，繰り返し計算により各頂点の重みを求めることで，それに対応する部品の評価値を定める [1], [2]．以下にその概略を示す．

部品評価値計算手順の概略

（1） ソフトウェアを実行し，動的支配関係を取得
（2） 動的支配関係を基に，動的部品グラフを生成
(部品評価値計算が必ず収束するようにするための補正を加えた擬似辺も含む [2].)
（3） 動的部品グラフの各頂点に初期重み (1/頂点数) を与える
（4） 動的部品グラフの各有向辺の重みを求める
(各頂点の重みを，その頂点から出ていく辺で分配する)
（5） 頂点に入ってくる辺の重みの総和を，その頂点の重みとして再定義する
（6） 頂点の重みが収束するまで，4,5 を繰り返し計算する
（7） 収束した頂点の重みを，その頂点に対応する部品の評価値として出力する

部品評価値計算例を， 図 2 に示す．部品 C_1，C_2，C_3 の重みは，それぞれ 0.4，0.2，0.4 で収束しており，この値を部品評価値として出力する．

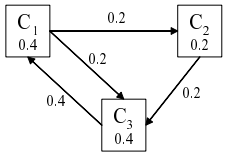

図 2 部品評価値計算例
Fig. 2 Example: Evaluation of Components

3. 評 価 実 験

2. 節の提案手法を基に，Java アプリケーションを対象とした部品評価値を計算する評価システムの実装を行った．このシステムでは，ソフトウェア部品の単位としてクラスとし，支配関係としてクラスの継承，インターフェースおよび抽象クラスの実装，メソッドの呼びだし，フィールド参照としている．

この実装システムを利用して，評価実験を行った．以降，実験結果ならびに考察について述べる．また，部品評価値のシーケンス図への適用について述べる．

3.1 実験結果及び考察

実装システムを利用して評価実験を行った． 表 1 は，javac(Java コンパイラ) に正常にコンパイルが終了するプログラムを引数に与えて実行した時の，部品評価値の上位 10 位までの結果である．

表 1 からわかるように，支配部品数の少ない部品であっても，部品評価値の高い部品が存在することがわかる．一般に，支配部品数が多い部品ほど，実行した機能に必要な，中心となる部品と考えられる．しかし，ある部品 A が支配部品数の多い部品 B のみを呼び出している場合，部品 B の呼び出し方法が記述されている部品 A も再利用にとって重要な部品であり，支配数が少ない部品でも，中心となる部品が存在しているはずである．提案手法を用いることで，機能実行における重要な部品を知ることができる．

次に，実行方法による部品評価値の違いを検証するため，j2sdk1.4.0_01.demo.jfc.Notepad プログラムに対し，複数機能を実行した時の，部品評価値の上位 10 位までを 表 2 に，ファイルオープン機能のみ実行したときの，部品評価値の上位 10 位までを 表 3 に示す．

表 1 javac：正常終了
Table 1 javac：Normal Execution
{(a): the number of the dominance relation
(b): Ranked by (a) }

評価値順位 Rank	クラス名 Classname	部品評価値 ($\times 10^8$) Value	支配部品数 (a)	支配数順位 (b)
1	v8.Main	6230715	17	11
2	v8.JavaCompiler	5401857	24	7
3	v8.comp.Gen	3308156	36	1
4	Main	2911060	2	53
5	v8.comp.Attr	2844601	34	2
6	v8.comp.Enter	2588095	28	3
7	v8.comp.TransInner	2007288	27	4
8	v8.code.ClassReader	1949322	26	5
9	v8.comp.Enter$CompleteEnter	1821256	16	14
10	v8.comp.TransTypes	1797646	26	5

表 2 Notepad：複数機能実行
Table 2 Notepad：Execution of Multiple Function
（See Table 1）

評価値順位 Rank	クラス名 Classname	部品評価値 ($\times 10^8$) Value	支配部品数 (a)	支配数順位 (b)
1	⟨System⟩	16207350	16	1
2	Notepad	13356511	12	2
3	Notepad$UndoAction	5732073	5	3
4	Notepad$OpenAction	5361253	4	4
5	Notepad$ShowElementTreeAction	5063501	3	6
6	Notepad$UndoHandler	4803297	3	6
7	ElementTreePanel	4740721	4	4
8	Notepad$3	4732369	3	6
9	Notepad$FileLoader	4731713	2	10
10	Notepad$2	4358630	2	10

表 3 Notepad：ファイルオープン機能実行
Table 3 Notepad：Execution of File-Open Function

評価値順位 Rank	クラス名 Classname	部品評価値 ($\times 10^8$) Value	支配部品数 (a)	支配数順位 (b)
1	Notepad	28475829	12	1
2	⟨System⟩	14920808	4	2
3	Notepad$OpenAction	10882916	3	3
4	Notepad$FileLoader	7715919	1	4
5	Notepad$ShowElementTreeAction	3800453	0	5
	Notepad$UndoHandler	3800453	0	5
	Notepad$UndoAction	3800453	0	5
	Notepad$NewAction	3800453	0	5
	Notepad$RedoAction	3800453	0	5
	Notepad$ExitAction	3800453	0	5

表 2 と 表 3 を比べてわかるように，実行方法によっ

て，部品評価値が異なることが確認できる．さらに，表 3 では，OpenAction クラスや FileLoader クラスといった，ファイルオープンに必要な部品の評価値が，表 2 よりも高いことがわかる．つまり，実行した機能に依存した部品評価値が得られることが確認できる．

3.2 シーケンス図への適用

部品評価値はどの部品が重要であるかの指標となり，再利用における部品選択の基準の一つとして利用できる．しかし部品評価値だけでは，部品が実際にどのように活用されているかという情報を取得することができない．そこで，本研究では，部品評価値の**シーケンス図**(Sequence Diagram) への適用を行う．

シーケンス図とは，オブジェクトの相互作用を時系列に沿って並べて表現した図のことをいう [4]．このシーケンス図を動的支配関係のみで作成すれば，全ての呼び出し関係が表示されるため，開発者にとってわかりにくい図となってしまう．そこでシーケンス図作成の際に，適当な部品評価値でフィルタリングし，重要な部品間の呼び出し関係のみを表示するようにする．フィルタリングすることにより，重要でない部品間の支配関係を削減することで，開発者が注目すべき部品，支配関係を絞り込めるため，ソフトウェア理解につながり，結果再利用の支援を可能とすると考えられる．

図 3 は， 表 3 の FileLoader クラスの部品評価値 (7715919) でフィルタリングを行ったシーケンス図である．全 19 クラスから成り立っていたシーケンス図が，3 クラス間の支配関係のみ出力されている．実際，この 3 クラスで，ファイルオープン機能は実装されており，全体を表示するより開発者に有益な表示となっている．

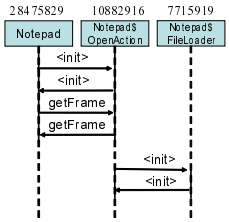

図 3 フィルタリングを行ったシーケンス図例
Fig. 3 Example: Filtered Sequence Diagram

4. まとめと今後の課題

本研究では，動的な情報を利用した部品評価手法の提案を行った．提案手法をシステムに実装し，評価実験を行った結果，特定の機能を実装している重要な部品の選択基準に利用できることを確認した．さらに，シーケンス図作成の際に，特定の部品評価値でフィルタリングすることで，ソフトウェア理解，再利用支援につながる情報の表示を行えることを確認した．

今後の課題として，さらに多くのアプリケーションに対して適用する，他の評価手法との比較を行う，有向辺によるフィルタリング，などが挙げられる．

文 献

[1] K. Inoue, R. Yokomori, H. Fujiwara, T. Yamamoto, M. Matsushita, S. Kusumoto: “Component Rank: Relative Significance Rank for Software Component Search”, ICSE2003(to appear). (2002).

[2] 藤原 晃, 横森 励士, 山本 哲男, 松下 誠, 楠本 真二 , 井上 克郎 “ソースコード間の関係を用いた再利用性評価手法の提案”, 情報処理学会研究報告, 2002-SE-136, Vol.2002, No.23, pp.155-162 (2002).

[3] C. Braun: Reuse, in John J. Marciniak: “Encyclopedia of Software Engineering”, Vol. 2, John Wiley & Sons, pp. 1055-1069 (1994).

[4] J. Rumbaugh, I. Jacobson, G. Booch : “The Unified Modeling Language Reference Manual”,Addison-Wesley Pub Co. (1998).

（平成 x 年 xx 月 xx 日受付）