ソフトウェア部品間の利用関係を用いた再利用性評価手法の提案

横森 励士
大阪大学大学院基礎工学研究科
yokomori@ics.es.osaka-u.ac.jp
藤原 晃
大阪大学大学院基礎工学研究科
h-fujiwr@ics.es.osaka-u.ac.jp
山本 哲男
大阪大学大学院基礎工学研究科
t-yamamt@ics.es.osaka-u.ac.jp
松下 誠
大阪大学大学院基礎工学研究科
matusita@ics.es.osaka-u.ac.jp
楠本 真二
大阪大学大学院基礎工学研究科
kusumoto@ics.es.osaka-u.ac.jp
井上 克郎
大阪大学大学院基礎工学研究科
inoue@ics.es.osaka-u.ac.jp

Abstract

再利用性の高い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，結果としてコストを削減できる．ソフトウェア部品の再利用性を評価する方法はこれまでに数多く提案されているが，その方法は全て，部品そのものの持つ静的な特性を計算して再利用性を評価するものである．しかし，現実に存在する部品に関しては，実際に多くのソフトウェア中に再利用されているという実績に基づいて再利用性を定量的に評価することが必要である．現実には，従来手法では再利用性が低いと評価されても，多くのシステムで再利用されているという部品は多く存在すると考えられる．
そこで，本論文では，利用実績に基づいたソフトウェア部品の再利用性評価手法について提案する．再利用性についての基本的な考え方は，以下の *(1)*~*(3)* である．*(1)* ソフトウェアを構成する部品間には相互に利用関係がある，*(2)* 一般に，時間が経過して多くのプロジェクト開発で再利用などが行われるに連れて部品の利用関係は変化していく，*(3)* 十分な時間が経過した状態のもとで，被利用数が多い部品は再利用性が高く，再利用性が高い部品から利用されている部品も再利用性が高い．
提案する手法に基づいて，プログラミング言語 *Java* で開発されたソフトウェア群から再利用性を計測するシステムを開発し，適用実験を行う．適用実験では，幾つかのソフトウェアシステムに適用し，本手法の有効性の検証，類似部品群を形成する際の閾値と相対的再利用性評価値との関係の検証を行う．実験結果では，実際に利用されている回数の多いクラスが上位を占め，本手法の有効性が確認された．

1. まえがき

ソフトウェアの大規模化と複雑化に伴い，高品質なソフトウェアを一定期間内に効率良く開発することが重要になってきている．これを実現するために様々な手法が提案されてきている．再利用はそれらの中でも最も有効なものの一つである．

再利用は既存のソフトウェア部品を同一システム内や他のシステムで用いることであると定義されている *[5]*．一般にソフトウェアの再利用は生産性と品質を改善し，結果としてコストを削減するといわれている．再利用を実際に用いる事でコストを削減する事ができた事例の報告も行われている *[4][9][13]*．

個々のソフトウェア部品の再利用性を評価する方法はいろいろ提案されている．*Etzkorn* らは，レガシーソフトウェア中の部品 *(C++*のクラス*)* に対して，様々なメトリクス値を計算し，それらの値を正規化して足し合わせることで，再利用性とすることを提案した *[7]*．また，山本らは，ソースコードが非開示なソフトウェア部品に対して，インタフェース部分の情報のみを用いて再利用性を評価する方法を提案している *[19]*．これらの方法は全て，部品そのものの持つ静的な特性を計算して再利用性を評価するものとなっている．また，提案された再利用性の評価値の妥当性については，複数の部品に対して得られた再利用性の値の順位と実際のプログラマによる主観的な再利用性の評価の結果が似ているといった評価が行われている．

しかし，現実に存在する部品に関しては，実際に多くのソフトウェア中に再利用されているという実績に基づいて再利用性を定量的に評価することが必要である．つまり，静的な特性や主観的な評価から再利用性が高いと判

断されても，実際の利用実績がなければ意味がないという立場である．現実には，従来手法では再利用性が低いと評価されても，多くのシステムで再利用されている部品は数多く存在すると考えられる．

そこで，本論文では，利用実績に基づいたソフトウェア部品の再利用性評価手法について提案する．再利用性についての基本的な考え方は，以下の *(1)*〜*(3)* の通りである．*(1)* ソフトウェアを構成する部品間には相互に利用関係がある．*(2)* 一般に，時間が経過し，多くのプロジェクト開発で再利用などが行われるにつれて部品の利用関係は変化していく．*(3)* 十分な時間が経過した状態のもとで，被利用数が多い部品は重要である（再利用性が高い）．また，重要な部品から利用されている部品も重要である．

このような実績に基づいた評価手法は，様々な分野において採用されている．例えば，計量社会学において，多くの論文誌に引用される論文誌は重要である，重要な論文誌に引用される論文誌はまた重要であるという再帰的な関係をもとに，各学術論文誌の重要度（*Influence Weight*）を評価する手法が提案されている *[15]*．また，サーチエンジン *Google* では，多くのページ良質なページからリンクされているページはやはり良質なページであるという再帰的な関係をもとに，あらゆるページの重要度（*PageRank*）を評価し，その結果に基づいて検索結果を表示している *[3, 16]*．

我々が提案する利用実績に基づいたソフトウェア部品の再利用性評価手法では，まず，評価対象のソフトウェア部品の集合に対して，それを構成する部品間の利用関係を抽出する．次に，各部品間で類似度が高い部品を集め，その部品群を一つの部品とみなして，部品群同士の利用関係を抽出する．この部品群同士の利用関係に基づいて，各部品群の再利用性評価値を求める．各部品群はそれぞれ再利用性評価値の重みを持ち，その評価値の重みに基づいて各部品群間の利用関係の重みが定義される．各部品群の再利用性評価値は，その部品群が利用される関係の重みの合計値となる．部品群の再利用性評価値を求める計算は，部品群間の利用関係を行列として定義する事で，行列の固有ベクトル計算に帰着される．部品群における再利用性評価値をもとに部品における再利用性評価値を決定する．

提案する手法に基づいて，プログラミング言語 *Java* で開発されたソフトウェア群から再利用性を計測するシステムを開発し，幾つかのソフトウェアシステムにも適用した．

以降，*2* 節では，本研究における再利用モデルの説明と相対的再利用性について述べる．*3* 節では，提案手法「R^3 法」の定義と計算方法について述べる．*4* 節では，相対的再利用性評価システム「*R^3-System*」の実装と *Java* ソースコードへの適用実験について述べる．*5* 節では，R^3 法の利用方法として，ソフトウェア部品検索への応用について述べる．

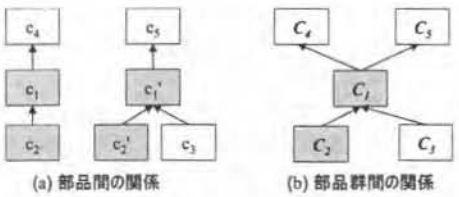

図 1. 類似した部品群

2. 再利用性評価モデル

ここでは，再利用に基づくソフトウェア開発をモデル化し，部品間の相対的再利用性について説明する．

2.1. ソフトウェア部品

一般にソフトウェア部品 *(Software Comopnent)* は再利用できるように設計された部品とされ，特に部品の内容を利用者が知る必要のないブラックボックス再利用ができるものを指すこともある *[1, 10]*．本論文ではより一般的に，ソースコードファイルやバイナリファイル，ドキュメントなどの種類を問わず，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．これらの部品間には互いに利用する，利用されるという利用関係が存在する．

2.2. 類似部品群

一般に，部品の集合には，コピーした部品や，コピーして一部を変更しただけの部品が数多く存在する．これらの部品を別々に扱ってしまうと，利用関係を正しく把握することができなくなる．コピーした部品間に存在する利用関係を把握する必要があるが，コピー元の特定は難しく，利用関係として定義するのは難しい．そこで，類似した部品をまとめることにより，部品の集合をいくつかの部品群に分類する．以降，類似した部品を集めた部品群を単に部品群と呼ぶ．

図 *1* を例に説明する．図 *1(a)* は部品間の利用関係である．部品 c_1 と c_1'，部品 c_2 と c_2' はそれぞれ類似した部品である．c_3, c_4, c_5 には類似性はない．部品 c_2 は c_1 を利用し，c_2', c_3 は c_1' を利用しているとする．また，c_1 は c_4 を，c_1' は c_5 をそれぞれ利用しているとする．

図 *1(b)* では，類似した部品をまとめて部品群としている．部品群に属する部品はそれぞれ
$C_1 = \{c_1, c_1'\}$，$C_2 = \{c_2, c_2'\}$，$C_3 = \{c_3\}$，
$C_4 = \{c_4\}$，$C_5 = \{c_5\}$ である．

部品群 C_i に属するある部品が，他の部品群 C_j に属する部品を利用している場合，その *2* つの部品群間 C_i，C_j

には利用関係があるとする．例えば，図 *1* では，c_1 と c_1' を利用する関係は，C_1 を利用する関係としてまとめる．また，c_1 と c_1' が利用している関係は，C_1 が利用している関係としてまとめる．

上述した考えを用いる場合には，*2* つの部品がどの程度類似しているか *(*類似度 *(Similarity))* を定量的に評価する必要がある．そのために類似度を評価するメトリクスを利用する．また，類似度に基づいてクラスタ分析を行い n 個の部品の集合を $m(1 \leq m \leq n)$ 個の部品群に分ける．類似度は *0* 以上 *1* 以下の範囲に正規化され，値が高いほど部品は良く類似しているとし，類似度 *1* を完全に部品が一致した場合（コピーした部品）とする．基準となる類似度の閾値 $t(0 \leq t \leq 1)$ を与え，部品群間の類似度が t 以下になるように分類する．

2.3. 相対的再利用性

個々の部品の再利用性を評価する手法は多く提案されている．*Etzkorn* らは，*Modularity, Interface Size, Documentation, Complexity* の *4* つの視点からオブジェクト指向ソフトウェアの再利用性を評価する手法を提案している．彼らはソースコードから計測される複数のメトリクスを正規化して足し合わせることで再利用性のメトリクスとし，*C++*のソースコードに対して実際にプログラマが評価した再利用性とメトリクス値を比較している *[7]*．

また，山本らはソースコードが非開示な部品に対して，そのインターフェイスから再利用性を評価する手法を提案している．彼らは理解容易性，利用容易性，テスト容易性，可搬性の *4* つの視点から再利用性メトリクスを定義し，*JavaBeans* を対象として実際にプログラマがアプリケーションを実装した結果とメトリクス値を比較している *[19]*．

これらの方法は全て，部品の構造やインターフェイスなど部品そのものの持つ静的な特性を計算して再利用性を評価するものとなっている．

これに対し本論文では，その部品がどの程度利用されているかという実績に基づいた評価を行うことを目的とする．この再利用性は多数の部品間の利用関係から相対的に決まるものであり，部品の静的特性から決まる再利用性とは区別して「相対的再利用性 *(Relative Reusability)*」と呼ぶ．

3. 提案手法

3.1. 概要

ソフトウェア開発者が，過去に開発されたソフトウェア部品を用いて新しいソフトウェアを開発する場合を想定する．一般に，開発者は過去に開発されたソフトウェア部品の中で，開発しようとしているソフトウェアに対して再利用性が高いと判断したものを用いると思われる．提案手法においては，この行為を新しいソフトウェア部品上における，過去に開発された部品への再利用性に関する支持投票と定義する．再利用によるソフトウェア開発が何度も繰り返されれば，再利用性の高い部品は何度も利用され，支持投票数が増加する．逆に再利用性の低い部品はあまり利用されず，したがって支持投票数は低い値に留まる．また，十分な時間が経過した状態で部品間の利用関係はある形に収束すると仮定する．このときソフトウェア部品は獲得した票数に応じた再利用性の評価値を持つと考えられるので，以下の式が成り立つ．

(部品の評価値) = (部品への投票数)

このとき，単純に獲得票数を数えるだけでなく，どのような部品から利用されたかによって票に重みづけをする．多くの部品から利用されるような優秀な部品（優秀な部品の開発者）に利用されている部品は，再利用性が高いとみなして，同じ一票の支持投票でも，再利用性の評価値の高い部品から投票された場合と，評価値の低い部品から投票された場合では，前者の票の方がより高い重みを持つようにする．

また，ある部品が利用している部品の数も考慮する．ある部品 *A* が多数の部品を利用している場合，*A* の各機能に占める各利用部品の割合が少なくなるので，再利用性の評価は分散してしまうとみなす．つまり，ある部品が複数の部品に投票している時は，票の重みは各利用部品にある配分率をもって配分され，以下の式が成り立つ．

*(*票の重み*)*
= *(*投票元の評価値*)* × *(*投票先に対する配分率*)*

このように，部品の集合において部品同士がお互いに再利用性を評価し，投票しあうことで決まっていく評価を「相対的再利用性 *(Relative Reusability)*」と呼び，部品が獲得した票の重みの合計値を「相対的再利用性評価値 (*Value of Relative Reusability*)」と呼ぶ．

実際に再利用を繰り返してソフトウェア開発を行う場合，新たに開発したソフトウェアが蓄積されていくため，部品の集合の要素数は単調増加し，利用関係は変化していく．相対的再利用性評価値は部品の集合における利用関係から相対的に求められる．そのため，部品の集合の要素数や利用関係が変化した場合に各部品の変化前後の評価値を直接比較してもあまり意味がない．

そこで，評価値そのものではなく，その評価値による部品の順位に着目する．部品の集合の要素数や再利用関係が変化したときも，その変化前後の部品の順位の変動を見ることで，部品の相対的再利用性がどのように変化したのかを理解することができる．

2.2 節で述べたように，実際の部品の集合には多数のコピーや類似部品が存在しているため，提案手法では部品群を部品の単位とし，部品群の相対的再利用性評価値による順位づけを行うことで評価を行う．この手法を「*Relative Reusability Ranking* 法 *(*R^3 法*)*」と呼ぶ．

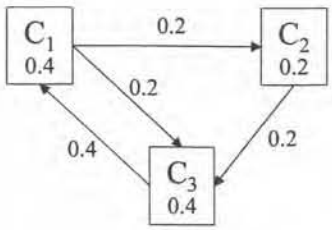

図 2. 再利用性評価の例

図 *2* を例に説明する．C_1, C_2, C_3 は部品群を表している．利用関係は利用する部品群から利用される部品群への矢印で表している．簡単のため，利用していない部品群への配分率は *0* としている．また，利用している部品には均等に配分する．C_1, C_2, C_3 の評価値を v_1, v_2, v_3 とし，評価値の総和が 1 となるように定義する．このとき，図 *2* の利用関係から各部品群の評価値は $v_1 = 0.4, v_2 = 0.2, v_3 = 0.4$ となる．よって C_1, C_3 は C_2 よりも相対的再利用性が高いと評価される．

3.2. 部品の分類

評価対象となる部品が n 個あるとし，それぞれ $c_1, c_2, ..., c_n$ とする．部品間には方向性を持つ利用関係があり，部品 c_i から c_j への関係を $r(c_i, c_j)$ と表したとき，

$$\begin{aligned} r(c_i, c_j) \quad = \quad & \textbf{if} \quad (c_i \text{は} c_j \text{を利用している}) \\ & \textbf{then} \quad \mathit{true} \\ & \textbf{else} \quad \mathit{false} \end{aligned}$$

と定義する．

また，部品間の類似度を $s(c_i, c_j)$ と表す．類似度は $0 \leq s(c_i, c_j) \leq 1$ に正規化されているとする．ここで，類似部品群を次のように定義する．

定義 *1* 分類の基準となる類似度の閾値を $t(0 \leq t \leq 1)$ とするとき，部品の集合 C を次の *(1)(2)* を満たすように分割した部分集合 $C_1, C_2, ..., C_m$ を類似部品群と呼ぶ．

(1) C_i に属するすべての部品について，類似度は t 以上であるような部品が同一部品群に存在する．

$$\forall c_k \in C_i \mid \exists c_l \in C_i \mid s(c_k, c_l) \geq t \tag{1}$$

(2) 異なる集合間の類似度は t より低い．
すなわち，すべての相異なる $i, j(1 \leq i, j \leq m)$ について次式が成り立つ．

$$\forall c_k \in C_i, \forall c_l \in C_j \mid s(c_k, c_l) < t \tag{2}$$

C を m 個の類似部品群に分割し，それぞれ $C_1, C_2, ..., C_m$ とする．以降，類似部品群を単に部品群と呼ぶ．部品群間には方向性を持つ利用関係があり，部品群 C_i から C_j への関係を $R(C_i, C_j)$ と表す．

定義 *2* $c_k \in C_i, c_l \in C_j$ とするとき，ある c_k, c_l について c_k から c_l への利用関係があれば，C_i から C_j への利用関係があるとみなし，次式で表す．

$$\begin{aligned} R(C_i, C_j) \quad = \quad & \textbf{if} \quad (\exists c_k, c_l | r(c_k, c_l)) \\ & \textbf{then} \quad \mathit{true} \\ & \textbf{else} \quad \mathit{false} \end{aligned} \tag{3}$$

3.3. 相対的再利用性評価値の定義

ここでは，相対的再利用性評価値について説明する．部品群 C_i の持つ相対的再利用性評価値を v_i と表し，C_i から C_j への利用関係の重みを w_{ij} と表す．また，部品 c_k の相対的再利用性評価値を v_{c_k} とする．

定義 *3* 部品群 C_i に属する部品 c_k の相対的再利用性評価値 v_{c_k} は v_i とする．つまり，同一部品群に含まれる類似した部品は同一評価値を持つ．

$$\forall c_k \in C_i | v_{c_k} = v_i \tag{4}$$

定義 *4* 部品群 C_i の相対的再利用性評価値は，部品群 C_i への利用関係の重み w_{ji} の総和である．

$$v_i = \sum_{j=1}^{m} w_{ji} \tag{5}$$

部品群 C_i から部品群 C_j への重みの配分率 *(Distribution Ratio)* を $d_{ij}(0 \leq d_{ij} \leq 1)$ と表す．

定義 *5* 部品群 C_i から C_j への利用関係の重み w_{ij} は，C_i の相対的再利用性評価値を配分率 d_{ij} で配分した値である．

$$w_{ij} = v_i d_{ij} \tag{6}$$

定義 *6* それぞれの部品群において，その部品群からの相対的再利用性評価値の配分率の和は *1* とする．

$$\sum_{j=1}^{m} d_{ij} = 1 \tag{7}$$

定義 *7* 利用している部品群への配分率は，利用していない部品群への配分率より高い．すなわち，
$R(C_i, C_j) = \mathit{true}, R(C_i, C_k) = \mathit{false}$ のとき，

$$d_{ij} > d_{ik} \tag{8}$$

とする．

3.4. 評価値の計算方法

ここでは，実際に相対的再利用性評価値を求める方法を説明する．

定義 *4, 5* より，次式が成り立つ．

$$v_i = \sum_{j=1}^{m} v_j d_{ji} \tag{9}$$

これを $v_i (i = 1, 2, ..., m)$ のすべてについて解けば，すべての部品群の評価値を求めることができる．

すなわち，

$$\begin{aligned} v_1 &= \sum_{j=1}^{m} v_j d_{j1} \\ v_2 &= \sum_{j=1}^{m} v_j d_{j2} \\ &\vdots \\ v_m &= \sum_{j=1}^{m} v_j d_{jm} \end{aligned} \tag{10}$$

という m 個の連立方程式を解けば良い．

これを行列の記法で表す．m 個の部品群の評価値を表す m 次元列ベクトルを V とする．

$$V = (v_1, v_2, ..., v_m)^t \quad (^t\text{は転置を表す})$$

また，C_i から C_j への配分率を表す $m \times m$ 行列を D とする．

$$D = \begin{pmatrix} d_{11} & d_{12} & \cdots & d_{1m} \\ d_{21} & d_{22} & \cdots & d_{2m} \\ \vdots & \vdots & \ddots & \vdots \\ d_{m1} & d_{m2} & \cdots & d_{mm} \end{pmatrix}$$

このとき連立方程式 *(10)* は

$$V = D^t V \tag{11}$$

と表される．

このとき，行列 D^t における固有値 $\lambda = 1$ に対応する固有ベクトルを V とすると，V は式 *(11)* を満たす．この固有ベクトル V を求めることで，相対的再利用性評価値を決定することができる．

3.5. 補正

前節までで定義した相対的再利用性評価値を実際のソフトウェア部品にそのまま適用すると，いくつかの問題が生じるため若干の補正が必要となる．以下ではその問題点と対策を説明する．

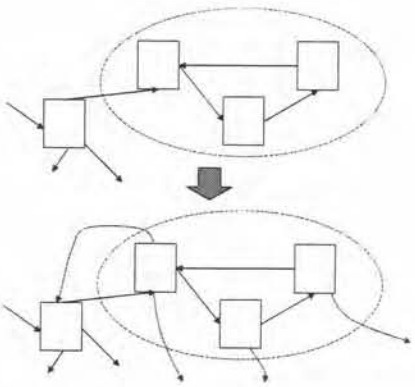

図 3. 評価が全体に循環しない場合

3.5.1 利用していない部品に対する評価

一般に，ソフトウェア開発においては，他の部品を一つも利用せずに開発した部品も存在する．

ある部品群 C_i が他のどの部品も利用していない場合，C_i はどの部品に対しても投票を行っていないことになる．したがってどの部品にも評価値を配分できないと考えて $d_{i0}, d_{i1}, d_{i2}, ..., d_{in}$ をすべて *0* とすると，定義 *6* を満たさないことになる．そこで，どの部品にも投票しない場合は「再利用性が非常に低い」という評価をすべての部品に対して投票したと解釈する．

補正 *1* 部品群 C_i がどの部品群も利用していない場合，すべての j について

$$d_{ij} = \frac{1}{m} \tag{12}$$

とする．

3.5.2 評価が全体に循環しない場合

図 *3* を例に説明する．ここで，四角は部品群を，矢印は利用関係を表している．図 *3* 上部では，楕円内の部品群に入る矢印は存在するが，楕円から出ていく矢印は存在しない．従って，再利用性評価の投票はこの楕円内に蓄積され，全体へ評価が循環しないことになる．本提案手法では，部品の集合における票の偏りを分析することで相対的再利用性を評価しているため，部品の集合全体に票が循環しなければ正しい評価を行なうことができない．そこで，部品群を利用しない場合には，非常に低い重みの票を投票すると考える．

補正 *2* 部品群のもつ評価値のうち $p(0 < p < 1)$ は利用した部品群にのみ配分し，$(1 - p)$ はすべての部品に配

分する．もとの配分率を d_{ij}，修正後の配分率を d'_{ij} とし，以下のように配分率を修正する．

$$d'_{ij} = pd_{ij} + (1-p)\frac{1}{m} \tag{13}$$

4. 再利用性評価システム

提案した再利用性評価モデルに基づいて，*Java* ソースコードを対象に相対的再利用性評価システム「R^3-*System*」を実装した．R^3 法を *Java* に適用する際の，モデルと *Java* の概念との対応を以下に示す．

部品　*Java* はオブジェクト指向言語であり，クラス単位での利用が行いやすい．また，原則として *1* つのソースファイルには *1* つのクラスを記述する．そこで，*Java* ソースコードファイルを部品の単位とする．

利用関係　部品間の利用関係をクラスの継承，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とする．

類似度　部品間の類似度を評価するメトリクスとして文献 *[18]* で提案されている，「S_{line}」を用いる．S_{line} は *2* つのソースコードファイル間で一致する行の割合で，*2* つのファイル間の類似度を表す．「*SMMT(Similarity Metrics Mesuring Tool)*」*[18]* は S_{line} をソースコードファイルから計測するシステムである．

4.1. システムの構成

R^3-*System* の構成図を図 *4* に示す．ここでは，N 個の *Java* ソースコードファイルを相対的再利用性の評価対象とした場合について述べる．

ファイル間の関係抽出部　N 個の *Java* ソースコードファイルを解析し，利用関係を抽出する．*Java* ソースコードの構文解析には，*ANTLR[2]* を利用している．

SMMT　*SMMT* を用いて，*Java* ソースコードファイル間の類似度 S_{line} を算出する．

クラスタ分析ツール　*SMMT* で得られた類似度をもとにクラスタ分析を行い，N 個の *Java* ソースコードファイルを M 個の部品群に分類する．クラスタ分析においては，分類の基準となる閾値 t をパラメータとして与える *(*定義 *1)*.

部品群間の関係抽出部　クラスタ分析ツールの結果とファイル間の関係抽出部の結果（部品間の関係）から，部品群間の関係を求める．

相対的再利用性計算部　部品群間の関係から，部品群に対して R^3 法で相対的再利用性評価値による順位づけを行う．なお，行列の固有ベクトル計算には，行列演算パッケージ *JAMA[11]* を利用している．

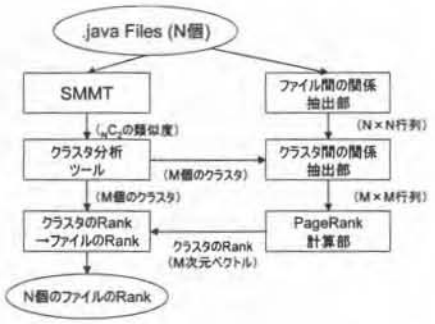

図 4. R^3 システムの構成図

部品群順位→ファイル順位変換部　M 個の部品群の相対的再利用性順位を，N 個の *Java* ソースコードファイルの順位に変換する．

4.2. 適用例

本提案手法では，*3* 節で説明したように再利用性評価値の順位によって各部品の相対的再利用性を評価している．そこで，部品の順位が利用実績を反映していることを示すために，実際のソフトウェア部品に対して R^3 法を適用する実験を行った．

実際に *Java* ソースコードに R^3 システムを適用した場合の適用例を以下に示す．評価対象として，*JDK-1.3.0* と我々の研究室内で開発された複数の *Java* アプリケーションの２つを選んだ．調整パラメータとして，定義 *1* で述べたクラスタ分析における分類の閾値を $t = 0.80$，補正 *2* で述べた票の重みの比率を $p = 0.85$ とした．

4.2.1 JDK-1.3.0 への適用

ここでは，評価対象として *Sun Microsystems [17]* から配布されている *Java 2 Software Development Kit, Standard Edition 1.3.0 (*以下 *JDK)* のソースコード *(1877* ファイル*)* を対象とした．*JDK* へ適用して得られた結果の一部を表 *1* に示す．

JDK は *Java* の基本パッケージであり，*Java* でアプリケーションを開発するときには *JDK* が必要となる．相対的再利用性の上位 *10* クラスについて見てみると，*Object, Class, Throwable* など，*Java* の言語仕様 *[8]* で利用しなければならないクラスが大半を占めている．*Java* 言語仕様によれば，*java.lang.Object* クラスはすべてのクラスのスーパークラスである．したがってすべてのクラスから利用されていることになり，相対的再利用性が *1* 位となっている．また，*java.lang.Class* クラスは実行中の

表 1. JDK-1.3.0 への適用結果

順位	クラス名	評価値
1	java.lang.Object	0.16126
2	java.lang.Class	0.08712
3	java.lang.Throwable	0.05510
4	java.lang.Exception	0.03103
5	java.io.IOException	0.01343
6	java.lang.StringBuffer	0.01214
7	java.lang.SecurityManager	0.01169
8	java.io.InputStream	0.01027
9	java.lang.reflect.Field	0.00948
10	java.lang.reflect.Constructor	0.00936
⋮	⋮	⋮
1256	sunw.util.EventListener	0.00011

クラスおよびインターフェイスを表すクラスで，このクラスを直接継承するようなクラスはないが，実行時のオブジェクト型の情報を取得するために頻繁に呼び出される．*java.lang.Throwable* クラスはすべてのエラーと例外のスーパークラスであり，したがって例外やエラーを扱うクラスはすべてこのクラスを間接的に利用していることになる．このように，実際に直接的，間接的に利用される回数の多いクラスが上位を占めている．また，最下位は *1256* 位で，該当するクラスは *622* あった．これらのクラスは，どのクラスにも利用されていない．

4.2.2 研究室内ソースコードへの適用

ここでは研究室内で過去に開発された *Java* アプリケーションのソースコードを適用対象とした *(582* ファイル*)*．

適用対象について簡単に説明する．

C-K メトリクス計測ツール 1　*Java* ソースコードから *C-K* メトリクス *[6]* を計測するツール．パッケージ名は *cktool*．*(29* ファイル*) ANTLR[2]* を使用してソースコードの構文解析を行っている．

C-K メトリクス計測ツール 2　上のツールを若干修正したもの．上と同じく *ANTLR* を用いている．パッケージ名は *cktool_new*．*(29* ファイル*)*

R^3-System　本論文で作成した *R^3-System*．ソースコードの構文解析には *ANTLR* を使用している．また，行列計算には *JAMA[11]* を使用している．そのほか *Caffe Cappuccino Class Library[14]* というライブラリも使用している．パッケージ名は *jp.ac.osaka_u.es.ics.iip_lab.metrics*．*(68* ファイル*)*

使用パッケージのソース　*ANTLR(*パッケージ名 *antlr, 188* ファイル*), JAMA(*パッケージ名 *Jama, 9* ファイル*), Caffe Cappuccino Class Library(*パッケージ名 *jp.gr.java_conf.keisuken, 245* ファイル*)* のソースコードも適用対象とした．

表 2. 研究室内ソースコードへの適用結果

順位	クラス名	評価値
1	antlr.Token	0.10727
2	antlr.debug.Event	0.06189
2	antlr.debug.NewLineEvent	0.06189
4	antlr.collections.impl.Vector	0.05434
5	jp.gr.java_conf.keisuken. text.html.HtmlParameter	0.05246
6	jp.gr.java_conf.keisuken. net.server.ServerProperties	0.03699
7	Jama.Matrix	0.01564
8	jp.gr.java_conf.keisuken. util.IntegerArray	0.01390
8	jp.gr.java_conf.keisuken. util.LongArray	0.01390
10	jp.ac.osaka_u.es.ics.iip_lab.metrics. parser.IdentifierInfo	0.01365
⋮	⋮	⋮
418	cktool_new.examples.Main	0.00050

その他　*(14* ファイル*)*

適用結果を表 *2* に示す．

研究室内で作成されたアプリケーションよりも，利用したパッケージの方が上位に来る傾向が見られる．また，主に機能を提供するようなクラスよりも，データを格納するようなクラスの方が高く評価される傾向にある．全体的に複数のアプリケーションで共通に使用されている *ANTLR* のクラスが高く評価されている傾向にある．これらの結果から，相対的再利用性評価値による順位づけは，実際の利用実績の傾向を反映していると考えられる．

また，*2* 位，*8* 位に同評価値の部品がそれぞれ存在している．これは，それぞれ二つのクラスが非常に類似していたため，同一部品群に存在していたからである．

4.3. 閾値 t の調整

ここでは類似度の閾値 t と相対的再利用性評価値との関係について検証する．ここでは，*JDK* に対して利用した部品への配分の比率を $p = 0.85$ と固定して t を変化させながら相対的再利用性評価値を計測する．t は 0.10 から 0.10 間隔で 1.00 まで変化させる．t の値が大きいほど類似部品群内の部品同士の類似度が高くなり，$t = 1.00$ では，内容が完全に一致する部品（コピーした部品）のみで構成される類似部品群になる．逆に t を 0 に近くすると，あまり類似していない部品も同一の類似部品群にまとめられる．$t = 0$ のときは，すべての部品が *1* つの類

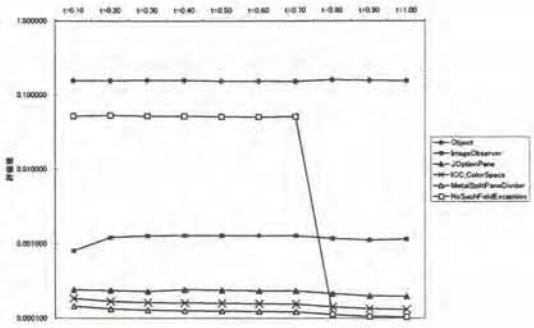

図 5. t と評価値の変化 ($p = 0.85$)

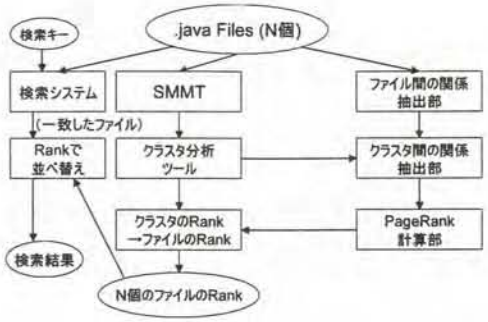

図 6. 検索システムの構成図

似部品群としてまとめられるため，相対的再利用性評価値を求めることができない．

上位 *10* 位と一部のクラスに関する評価値の順位を表 *3* に示す．表 *3* は，$t = 0.80$ での評価値および順位で整列している．上位 *10* 位については，*Object* クラスなどには変化はないが，$t = 0.70$ と $t = 0.80$ を境に *Throwable* と *Exception* の順位が大幅に入れ替わっている．これは，$t = 0.70$ 以下で分類した場合には *Exception* クラスを含めて *62* 個の例外クラスが *1* つの類似部品群にまとまることが原因である．そのため，*Throwable* や *StringBuffer* など t が *0.80* 以上の時には上位にあったクラスが *65* 位以下に落ちている．

t の変化によって評価値がどのように変化するかをグラフにしたものが図 *5* である．横軸は t の値，縦軸は評価値として変化の様子を示している．縦軸の目盛は常用対数で表示している．*NoSuchFieldException* は *Exception* のサブクラスである．グラフからも，$t = 0.70$ と $t = 0.80$ を境に例外クラスの評価値が大きく変化していることが明らかである．例外クラス以外では，$t = 0.10$ から $t = 0.90$ の範囲では大きな変化は見られない．

表 *4* は t の値ごとの *Spearman* の順位相関係数を示している．順位相関係数は 0.70 以上と全体的に高く，t の差が 0.10 の場合には，順位相関係数は 1 に近くなっている．上述した *Exception* についての問題はあるが，t が変化しても全体的な順位の傾向はあまり変動しないことが分かる．

以上の結果から，*JDK* に対する解析においては，閾値 t は例外クラスに関しては影響を与えうるが，その他のクラスに対しては影響が薄い事がわかる．$t = 0.70$ 以下で分類するかどうか，すなわち例外クラスをすべて同一の類似部品群として扱うかどうかは，対象とする部品のドメインや評価の目的によって適宜判断する必要がある．また例外クラスを評価対象に入れるかどうかも本提案手法を適用する際には検討すべきであると考えられる．

5. 部品検索システムへの応用

ここで提案した R^3 法の応用例として，ソフトウェア部品検索が考えられる．これは，インターネットを通じてソフトウェア部品を収集し，収集した部品に対してあらかじめ R^3 法で部品を順位付けする．開発者は再利用したい部品を検索キーを用いて検索し，システム側は，検索した結果を相対的再利用性で順位付けして表示する．これにより，開発者は再利用実績の高い部品を容易に入手し，再利用することができる．

5.1. 部品検索システム

図 *6* は部品検索システムを表し，R^3 システムである図 *4* から追加した部分の説明を以下に述べる．

検索システム　検索キーが入力され，検索キーに一致する部分を持つソースコードファイルを返す．

検索結果の順位づけ部　相対的再利用性の順位をもとに，検索結果を順位づけして表示する．

5.2. 検索の例

ここでは検索の例として，*4* 節の適用例で挙げたファイル群に，*[12]* において一般に公開されているソースコードファイルを追加した *3131* ファイルに対する検索について説明する．追加したソースファイルは，テキストエディタ *JEDIT*，ペイントソフト *JHotDraw*，プロジェクトのスケジューリングツール *JProjectTimer* に関するソースファイルなどである．

ファイルからの入力時に利用される，*FileInputStream* をキーとして *grep* を用いて検索を行った．ただし，コメントにのみ現れる場合は除外している．検索の結果 *47* のクラスが該当し，相対的再利用性の順位をもとに検索結果を順位づけしたところ，表 *5* のようになった．*FileInputStream*

表 3. t と順位の変化 ($p = 0.85$)

クラス名	t=0.10	t=0.20	t=0.30	t=0.40	t=0.50	t=0.60	t=0.70	t=0.80	t=0.90	t=1.00
java.lang.Object	1	1	1	1	1	1	1	1	1	1
java.lang.Class	2	2	2	2	2	2	2	2	2	2
java.lang.Throwable	66	66	66	66	66	66	65	3	3	3
java.lang.Exception	3	3	3	3	3	3	3	4	4	4
java.io.IOException	3	3	3	3	3	3	3	5	6	6
java.lang.StringBuffer	68	68	68	68	68	68	67	6	7	7
java.lang.SecurityManager	69	69	69	69	69	69	68	7	8	8
java.io.InputStream	72	71	71	71	71	71	70	8	9	9
java.lang.reflect.Field	73	72	72	72	72	72	71	9	10	10
java.lang.reflect.Constructor	74	73	73	73	73	73	72	10	11	11
⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮
java.awt.image.ImageObserver	316	191	172	172	166	166	163	103	101	94
⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮
javax.swing.JOptionPane	832	671	637	578	563	562	561	499	496	497
⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮
java.awt.color.ICC_ColorSpace	1046	961	941	931	923	917	914	850	828	833
⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮
javax.swing.plaf.metal.MetalSplitPaneDivider	1403	1343	1326	1318	1315	1314	1306	1253	1229	1228

表 4. 順位の相関 ($p = 0.85$)

	t=0.10	t=0.20	t=0.30	t=0.40	t=0.50	t=0.60	t=0.70	t=0.80	t=0.90	t=1.00
t=0.10	1.000	0.880	0.849	0.837	0.830	0.828	0.823	0.730	0.713	0.710
t=0.20	0.880	1.000	0.968	0.958	0.951	0.949	0.943	0.845	0.824	0.821
t=0.30	0.849	0.968	1.000	0.991	0.984	0.982	0.976	0.877	0.855	0.852
t=0.40	0.837	0.958	0.991	1.000	0.993	0.991	0.985	0.886	0.863	0.861
t=0.50	0.830	0.951	0.984	0.993	1.000	0.998	0.995	0.895	0.872	0.870
t=0.60	0.828	0.949	0.982	0.991	0.998	1.000	0.996	0.897	0.874	0.871
t=0.70	0.823	0.943	0.976	0.985	0.995	0.996	1.000	0.901	0.878	0.875
t=0.80	0.730	0.845	0.877	0.886	0.895	0.897	0.901	1.000	0.976	0.973
t=0.90	0.713	0.824	0.855	0.863	0.872	0.874	0.878	0.976	1.000	0.997
t=1.00	0.710	0.821	0.852	0.861	0.870	0.871	0.875	0.973	0.997	1.000

を実現しているクラスが *2* 位となり，その他のクラスは利用例である．上位にあがったクラスの利用方法を見てみると，設定ファイルからのプロパティの抽出とファイルのデータ列の抽出に関するものが大半で，残りは圧縮ファイルの内容一覧の抽出，構文解析であった．一方で，下位のクラスでは，上位のクラスのような利用例も存在するが，*Java* で用意されている例外機構を無効化するためのラッパークラスなど，特殊な利用例も数多く存在した．

全体的な傾向として，一般的な利用例は上位・中位・下位に満遍なく存在しているが，特殊な利用例は下位に集中していた．そのため，本手法を用いた検索は再利用に適していないプログラムを除外するのに有効な手法であると思われる．個々の利用目的に関して検索を行う場合には，検索キーの追加などを行う事で十分対応できると思われる．

また，*1* 位の *java.lang.Package* は利用例のクラスである．利用例としては *3* 位以下のクラスとあまり大差がないにもかかわらず，*FileInputStream* を実現しているクラスより評価値がはるかに大きく，順位が上である．これは，*java.lang.Package* は *Java* パッケージの実装に関するクラスで，他のクラスから頻繁に利用されるためである．

相対的再利用性評価値による順位づけは，実際の利用実績の傾向を反映しているため，再利用を目的とした検索にも応用できると考えられる．しかし，*Java* の言語仕様で利用しなければならないクラスがたまたま利用していた場合に，そのクラスが上位に入ってしまうという問題点がある．実際に提案手法を適用し検索システムを実現する際には，対象とする部品のドメインや評価の目的によって，利用関係の重みづけの変更や「言語仕様上利用しなければならないクラス」の解析からの除外を検討すべきであると考えられる．

6. まとめ

本論文では，ソフトウェア部品の相対的再利用性を定義し，その評価方法として R^3 法を提案した．実際に R^3 法に基づいて相対的再利用性を評価するシステム *R^3-System* を実装し，幾つかのソフトウェアシステムに適用

表 5. FileInputStream の検索結果

順位	クラス名	評価値
1	java.lang.Package	0.007389
2	java.io.FileInputStream.java	0.000775
3	java.awt.color.ICC_Profile	0.000605
4	jp.gr.java_conf.keisuken.net.server.ServerProperties	0.000600
5	java.awt.Cursor	0.000509
6	java.awt.Toolkit	0.000282
7	cktool_new.examples.oofp	0.000171
7	cktool.examples.oofp	0.000171
9	jp.ac.osaka_u.es.ics.iip_lab.metrics.component.java.ClassNameDAO	0.000120
9	jp.ac.osaka_u.es.ics.iip_lab.metrics.component.java.JavaFileDAO	0.000120

した．いくつかの例に適用した結果、一般的な使い方をしているクラスが集中して上位のランクに現れた。このように、相対的再利用性は、ソフトウェア部品の検索のための順位付けに有用なものと思われる。

再利用によるソフトウェア開発において，多くのソフトウェア部品の中から開発者が本当に必要としているものを見つけ出すのは困難な作業であるが，再利用する部品を選択する基準として R^3 法を用いることで開発者の負担を軽減できると考えられる．

今後の課題としては，さらに多くの部品への適用，*Java* ソースファイルにおける継承，実装，メソッド呼びだしの各関係の重みづけの検討，部品検索システムの検討と実装などが挙げられる．

謝辞　本研究は，平成 *13* 年度科学技術振興事業団計算科学技術活用型特定研究開発推進事業 *(ACT-JST)* の支援を受けている．

参考文献

[1] 青山, 中所, 向山: コンポーネントウェア, 共立出版, *(1998).*

[2] *"ANTLR Website", http://www.antlr.org/*

[3] 馬場: *"Google* の秘密 *- PageRank* 徹底解説*", http://www.kusastro.kyoto-u.ac.jp/ baba/wais/pagerank.htm*

[4] *V. R. Basili, G. Caldiera, F. McGarry, R. Pajerski, G. Page and S. Waligora: "The software engineering laboratory - an operational software experience",* Proc. of ICSE14, *pp. 370-381 (1992).*

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

[6] *S. R. Chidamber and C. F. Kemerer: "A metrics suite for object-oriented design",* IEEE Trans. on Software Eng., *Vol.20, No.6, pp.476-493 (1994).*

[7] *L. H. Etzkorn, W. E. Huges Jr., C. G. Davis: "Automated reusability quality analysis of OO legacy software",* Information and Software Technology, *Vol. 43, Issue 5, pp. 295-308 (2001).*

[8] *J. Gosling, B. Joy, G. Steele and G. Bracha: "The Java Language Spwcification Second Edition", Sun Microsystems, http://java.sun.com/docs/books/jls/ second_edition/html/j.title.doc.html*

[9] *S. Isoda: "Experience report on a software reuse project: Its structure, activities, and statistical results",* Proc. of ICSE14, *pp.320-326 (1992).*

[10] *I. Jacobson, M. Griss and P. Jonsson: Software Reuse,* Addison Wesley, *(1997).*

[11] *"JAMA : A Java Matrix Package", http://math.nist.gov/javanumerics/jama/*

[12] *"SourceForge", http://sourceforge.net/*

[13] *B. Keepence and M. Mannion: "Using patterns to model variability in product families",* IEEE Software, *Vol. 16, No. 4, pp. 102-108 (1999).*

[14] 西本: *"Java* 言語について*", http://cappuccino.ne.jp/keisuken/java/*

[15] *G. Pinski and Francis Narin: "Citation influence for journal aggregates of scientific publications: Theory, with application to the literature of physics", in Information Processing and Management, Vol. 12, Num 5, pp. 297-312, (1976).*

[16] *L. Page, S. Brin, R. Motwani, T. Winograd: "The PageRank Citation Ranking: Bringing Order to the Web", http://www-db.stanford.edu/ backrub/pageranksub.ps*

[17] *"The Source For Java(TM) Technology", Sun Microsystems, http://java.sun.com/*

[18] 山本, 松下, 神谷, 井上: *"*ソフトウェアシステムの類似度とその計測ツール *SMMT",* 電子情報通信学会論文誌 *D-I(*採録決定*).*

[19] 山本, 鷲崎, 深澤: "再利用特性に基づくコンポーネントメトリクスの提案と検証", ソフトウェア工学の基礎ワークショップ *(FOSE2001), (2001).*