論　文

利用実績に基づくソフトウェア部品重要度評価システム

横森　励士[†]　藤原　晃[†]　山本　哲男[††]　松下　誠[†††]
楠本　真二[†††]　井上　克郎[†††]

The Importance Evaluation System of Software Components Based on the Use Frequency

Reishi YOKOMORI[†], Hikaru FUJIWARA[†], Tetsuo YAMAMOTO[††],
Makoto MATSUSHITA[†††], Shinji KUSUMOTO[†††], and Katsuro INOUE[†††]

あらまし　近年，大量の大規模ソフトウェアが開発されており，開発現場ではソフトウェアがよく再利用されているが，現在のプログラム開発環境では個々の開発者の知識に頼っているのが現状で，知識の共有が満足になされていない．本論文では，ソフトウェア部品の検索において部品の選別に利用するための手法として，ソフトウェア部品の利用実績に基づいて各部品の重要度を測定し，順位付けする手法を提案する．この手法では，実際に開発された複数のシステム内に存在する部品に対して，利用関係に基づいてグラフ及び行列を構築し，繰り返し計算を行うことで，重要度を測定する．また，提案手法に基づき，Java で開発されたソフトウェア群から各部品の重要度を計測するシステムを開発し，適用実験を行う．実験結果では，実際に利用される回数の多いクラスや重要な機能をもつクラスが上位を占め，本手法で測定した部品の重要度が利用実績に基づいた定量的な指標であることを確認できた．この手法を部品検索において利用することで，利用実績のある部品を効果的に取得できる．

キーワード　ソフトウェア部品，部品評価，利用実績，Java

1. まえがき

近年，大規模で複雑なプログラムを含む大量のソフトウェアが開発され，様々な場所で様々な目的で利用されている．これらのプログラムの中には似た種類のプログラムが多数存在し，開発期間の短縮や品質向上を目的として，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利用する，いわゆるソフトウェアがよく再利用されている．ソフトウェアの再利用による効果を最大限に引き出すためには，開発者が今から開発しようとするソフトウェアに必要な部品及びライブラリに関する知識をもつことが重要になってくるが，知識の共有が満足になされていないために，同種のプログラムが別々の場所で，独立して開発されていることも多い．

一方でインターネットの普及により，SourceForge [21] などのソフトウェア開発に関する情報を交換するコミュニティが誕生し，大量のプログラムソースコードが簡単に入手できるようになった．これらの公開されている大量の部品の中から，開発者の必要としている機能をもつ部品，その機能の使い方を示している部品のような，再利用に有益な情報を提供する検索システムを実現することで，知識の共有が実現でき，再利用を促進することができると考えられる．

知識の共有を目的としたソフトウェアの部品検索システムを構築する際，検索された部品を評価し順位付けし，選別して表示する仕組みが必要となる．部品の特性から個々のソフトウェア部品の再利用性を評価する手法が提案されているが [4], [17]，このような部品検索システムにおける利用を考えた場合，目的に合った部品を選出することと同様に，いろいろなソフトウェアで使われ完成度が高い部品を選出することが必要であると考えられる．このとき，各部品の利用実績を定

[†] 大阪大学大学院基礎工学研究科，豊中市
Graduate School of Engineering Science, Osaka University, Toyonaka-shi, 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, 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, Toyonaka-shi, 560–8531 Japan

量的に評価した指標が必要となる．

そこで本論文では，ソフトウェア部品の利用実績に基づいて各部品の重要度を測定し，順位付けし，評価する手法（Component Rank 法，CR 法）を提案する．この手法では，十分な時間が経過し利用関係が収束したソフトウェア部品の集合に対して，各ソフトウェア部品間に存在する利用関係に基づいてグラフ及び行列を構築し，構築された行列に対して繰り返し計算を行うことで各部品の重要度を測定する．求められる重要度は，開発者が利用関係に沿って参照を行うと仮定した場合の各部品の参照されやすさを表しており，よく利用される部品や，重要な部品から利用される部品の重要度は高くなる．

また，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重要度を計測するシステムを開発し，求めた値が妥当であるかを検証するためにいくつかのソフトウェアシステムに適用する．また，重要度をもとに検索された部品を評価し順位付けし，選別して表示する部品検索システム（Software Product Archiving, analyzing, and Retrieving System, SPARS）について説明する．

以降，**2.** では，提案手法「CR 法」の定義と計算方法について述べる．**3.** では，部品重要度評価システム「CR-システム」の実現について述べる．**4.** では，Java ソースコードへの適用実験について述べる．**5.** では，関連研究及びソフトウェア部品検索への考察について述べる．最後に **6.** では，まとめと今後の課題について述べる．

2. Component Rank 法

ここでは，部品の概念をモデル化し，その上で利用実績に基づいた部品重要度の評価手法（Component Rank 法，CR 法）について説明する．

2.1 部品と部品間の利用関係

一般にソフトウェア部品（Software Component）とは再利用できるように設計された部品とされている [7], [12]．ここではより一般的に，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．本論文では各部品を頂点，部品間の利用関係を利用する側からされる側への有向辺として，グラフ上に表現する．以下，この部品間の利用関係を表現したグラフを部品グラフ（Component Graph）と呼ぶ．以下では，V を部品（頂点）の集合，E を有向辺の集合として，$G = (V, E)$ と表現する．

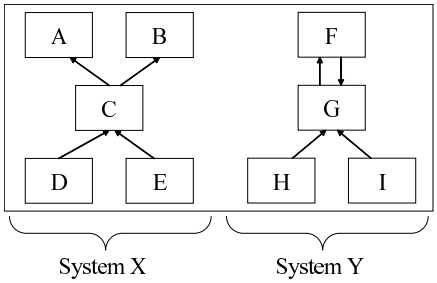


図 1　部品グラフの例
Fig. 1　An example of component graph.

図 1 は二つのソフトウェアシステム X と Y を部品グラフ化したものである．X は A から E の五つ，Y は F から I の四つの部品で構成されており，部品 C は部品 A 及び B を利用し，部品 D 及び E は部品 C を利用している．同様に部品 H と I は部品 G を利用し，部品 G と F はお互いに利用し合っている．

2.2 重みの定義

CR 法では，部品グラフ $G = (V, E)$ 上の個々の辺及び頂点に対して重みを計算し，対応する頂点の重みをもとに各部品の重要度を評価する．最初に，頂点の重みを次のように定義する．

［定義 1］（頂点の重みの和） 部品グラフ G 上の各頂点 v は $0 \leq w(v) \leq 1$ の重みをもち，G の頂点の重みの総和は，1 とする．

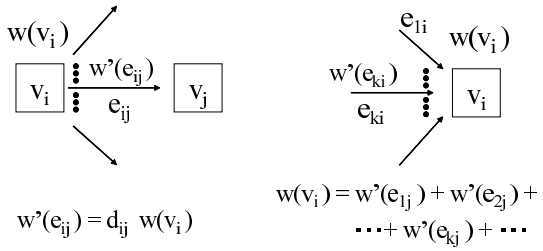
$$\sum_{v \in G} w(v) = 1$$

また，部品グラフ G 中の頂点 v の重みを求めるために，辺の重みを次のように定義する．

［定義 2］（辺の重み） 頂点 v_i から v_j への辺 e_{ij} に関する辺の重み $w'(e_{ij})$ を次のように定義する．

$$w'(e_{ij}) = d_{ij} \times w(v_i)$$

図 2 (a) はこの定義を図示したものである．d_{ij} は配分率と呼び，$0 \leq d_{ij} \leq 1$ かつ $\sum_i d_{ij} = 1$ を満たす値とする．頂点 v_i から v_j へ利用関係が存在しない場合，ここでは $d_{ij} = 0$ とする．この配分率 d_{ij} は，次に示す頂点の重みの定義において，有向辺の終点となる頂点の重みの決定に利用される．図 2 (b) は次の定義を図示したものである．



(a) Weight of edge (b) Weight of node

図 2 重みの定義
Fig. 2 Definition of weights.

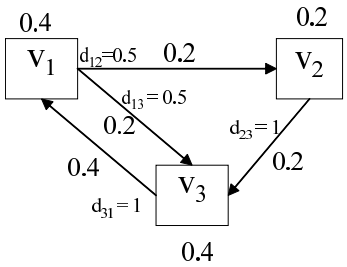


図 3 重みの計算例
Fig. 3 An example of stable weights assigned to nodes and edges.

［定義 3］（頂点の重み） $\mathrm{IN}(v_i)$ を v_i を終点とする有向辺の集合とする．このとき，頂点 v_i の重みは v_i が終点となる有向辺 e_{ki} の重みの総和とする．つまり，

$$w(v_i) = \sum_{e_{ki} \in \mathrm{IN}(v_i)} w'(e_{ki})$$

2.3 重みの計算

2.2 の重みの定義に基づいて，頂点 $w(v_i)$ に対して次の方程式が生成できる．

$$w(v_i) = \sum_{e_{ki} \in \mathrm{IN}(v_i)} d_{ki} \times w(v_k)$$

各頂点に関してこの方程式を立てることで，$n(=|V|)$ 個の連立方程式が生成できる．また，各頂点の重みをベクトル W として次のように表現し，

$$W = \begin{pmatrix} w(v_1) \\ w(v_2) \\ \vdots \\ w(v_n) \end{pmatrix}$$

配分率を次の行列 D として表現することで，

$$D = \begin{pmatrix} d_{11} & d_{12} & \ldots & d_{1n} \\ d_{21} & d_{22} & \ldots & d_{2n} \\ \vdots & \vdots & \ddots & \vdots \\ d_{n1} & d_{n2} & \ldots & d_{nn} \end{pmatrix}$$

$n(=|V|)$ 個の連立方程式を次のように表すことができる．ここで行列 D^t は D の転置行列を表す．

$$W = D^t W \qquad (1)$$

定義 1 から，行列 D^t は推移確率行列の性質を満たしているため，D^t を用いて開発者が部品をどのように参照するかをマルコフ連鎖 [2] でモデル化し表現する．すなわち，開発者はある部品を参照した後に，辺に沿って利用関係のある部品の一つを参照するとみなす．ここで，式 (1) を満たす解 W は D^t に関する定常分布となり，対象とする部品の集合の中での参照を十分長い間時間繰り返した場合の，開発者によって各部品が参照されている確率を示すことになる．つまり，重みが高い部品ほど開発者によって頻繁に参照されることになる．この値を重要度とすることで，ただ単に利用数が多い部品だけでなく，利用数が多い部品が利用している部品も重要であると評価することができる．定常分布が一意に求められることを示すためには，D^t が既約であることが必要となるが，CR 法では，既約であることを保証するために後述する補正を加えている．

この場合，D^t における絶対値最大の固有ベクトルを求めることで，定常分布を求めることができるが，行列 D^t の絶対値最大の固有値は 1 であるため，累乗法（power method）を用いて適当な初期ベクトルに対して繰り返し D^t を掛けることで，近似解を容易に求めることができる．図 3 は，与えられたグラフにおいて各頂点の重み（重要度）を計算した結果である．v_1 は二つの有向辺の始点で，v_1 の重み 0.4 は二つの辺に 0.2 ずつ等分されている（つまり，$d_{12} = d_{13} = 0.5$）．また，v_3 は二つの有向辺の終点で，それぞれの辺が 0.2 の重みをもつため，v_3 の重みは 0.4 であることがわかる．

2.4 部品重要度計算に関する補正

前節で部品の重要度の計算に関する説明を行ったが，実際に運用する場合不具合が起こる場合がある．例えば，部品 i がどの部品も利用していない場合，頂点 v_i からすべての頂点への配分率 d_{ix} がすべて 0 になり，配分率に関する定義（頂点からの辺への配分率の総和

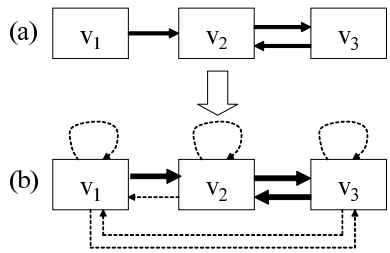


図 4　部品重要度計算に関する補正（擬似辺の追加）
Fig. 4　Introduction of pseudo use relation for convergence.

が 1）を満たさなくなる．また，グラフが図 4 (a) のように強連結でない場合，頂点 v_1 の重みが 0 になってしまい，v_1 から v_2 への利用関係を正しく評価することができない．

このような場合，与えられた行列が既約であることを保証できず，定常分布が一意に定まらない場合がある．そこで，すべての頂点を図 4 (b) のように低い配分率の擬似辺で結ぶことで，与えられたグラフを強連結なグラフに変換し，行列が既約であることを保証する．この擬似辺は，各部品における自分を含めたすべての部品への明示的でない参照を意図している．ここで p は実際の辺と擬似辺の重みの配分比率を指す．

［定義 4］（修正配分率）　頂点 v_i を始点とする辺への配分率 $d'(e_{ij})$ を次のように定義する．

$$d'_{ij} = \begin{cases} p \times d_{ij} + (1-p)/n & \text{if } v_i \text{ からの辺が存在} \\ 1/n & \text{if } v_i \text{ からの辺がない} \end{cases}$$

この補正は，前節で説明したマルコフ連鎖を用いたモデル上での参照行為を，「ある部品を参照した後に，辺に沿って利用関係のある部品の一つを参照するが，ある確率（$1-p$）で，ランダムに全部品の中の一つを参照する」のように修正する．この修正により，参照行為をより自然な形でモデル化することができる．

2.5　部品のグループ化

システムを構築する際には，過去に開発したシステムの部品の一部を再利用して開発が行われることがよくある．これらの部品は，ライブラリとして再利用されるほかに，コピーされたり，コピーされた上で大小様々な変更が加わった上で再利用されることが考えられる．そのため，様々なソフトウェアから部品を集めた場合，その部品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（類似部品）が数多く存在すると考えられる．

このとき，異なるシステムをまたいで類似部品が現れる場合を考える．再利用という観点から考慮すると，これらの類似部品は類似部品中のある一つの部品を利用してコピーして作成されたと推測することができる．そのため，それらの類似部品間に利用関係を定義するのが妥当であると考えられる．ライブラリとして再利用された場合には，部品グラフ上で利用関係を有向辺として定義できるが，コピーの場合はコピー元の特定が難しく，部品グラフ上の有向辺として定義するのは難しい．

そこで，提案手法では似た部品をグループ化し，部品群それぞれに対して重要度を与える．このとき各部品の重要度は，その部品が属する部品群の重要度とする．その際，それぞれの類似部品への辺が，グループ化を行うことで一つの部品群への辺とみなされる．そのため，コピーして再利用される回数が多いほど，その部品群はたくさんの部品から利用されることになる．グループ化によりコピーされた部品の重要度が高くなるため，コピーされたという利用実績を重要度に反映させることができる．

グループ化を行う際には，二つの部品がどの程度類似しているかを定量的に評価するために部品間の類似度（Similarity）を評価するメトリクスを利用する．以下，部品 v_1，v_2 間の類似度を $s(v_1, v_2)$ と表す．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よ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．また，グループ化のしきい値として t という値を定義し，$s(v_1, v_2) \geq t$ であるときに $s(v_1, v_2)$ は類似関係にあるとする．部品集合 V 内で類似関係にある部品を同一の部品群とすることで，V の商集合からなる類似部品群の集合 V' を求める．このとき，G の部品群グラフ（clustered component graph）$G' = (V', E')$ を次のように定義する．

［定義 5］（部品群グラフ $G' = (V', E')$）　V の商集合からなる集合 V' を G' の頂点集合とする．また v_i，v_j が属する V' 中の集合をそれぞれ v'_i，v'_j としたときに，$E' = \{(v'_i, v'_j) | (v_i, v_j) \in E\}$ を G' の辺集合とする．

図 5 は部品のグループ化を行った際に，コピーされた部品の重要度がどのように変わるかを例で示したものである．三つのシステムのうち，二つのシステムには $X.A$ 及び $Y.A$ という同一の部品が存在する．A という部品がライブラリとして再利用された場合は，部

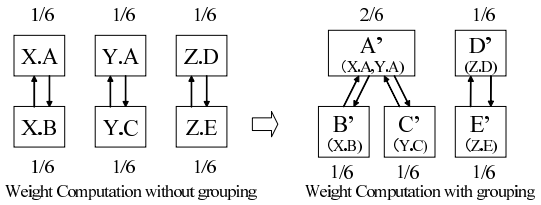


図 5　グループ化の効果
Fig. 5　Effect of grouping components.

品 A が部品 $X.B$ 及び $Y.C$ の両方と利用関係にあることを把握できる．しかし一方で，部品がコピーされて再利用された場合は，部品が属する名前空間（パッケージ名）が異なる，または部品名が異なるなどの理由から $X.A$，$Y.A$ は別々の部品として扱われてしまう．このとき，グループ化を行わないと，コピーされた部品は別々の部品として評価されるため，図 5 の場合すべての部品の重要度が等しくなる．

そこで，提案手法では部品間の類似度を測定することでグループ化を行う．これにより $X.A$ 及び $Y.A$ が同一部品であると判定され，部品 $X.A$ 及び $Y.A$ へのそれぞれの有向辺が一つの部品群 A' への有向辺に統合されるため，コピーが存在する部品 A の重要度が他の部品よりも高くなることがわかる．以降の実際の重要度計算では，部品グラフではなく部品群グラフを対象としており，部品群グラフに対して修正配分率に関する補正を行った上で部品群における重要度を計算している．

3. Component Rank システム

2. で提案した CR 法に基づいて，我々は Java プログラムを対象としてソースコードの重要度を評価するシステム「Component Rank システム,（以下 CR-システム）」を構築した．ここでは，構築した CR-システムについて説明する．

3.1 CR-システムの実現

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］ オブジェクト指向言語 Java は原則として一つのソースファイルには一つのクラスを記述している．そこで，クラス単位での利用が行いやすい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，インタフェース及び抽象クラスの実装，メソッドの呼出し，フィールド参照とする．

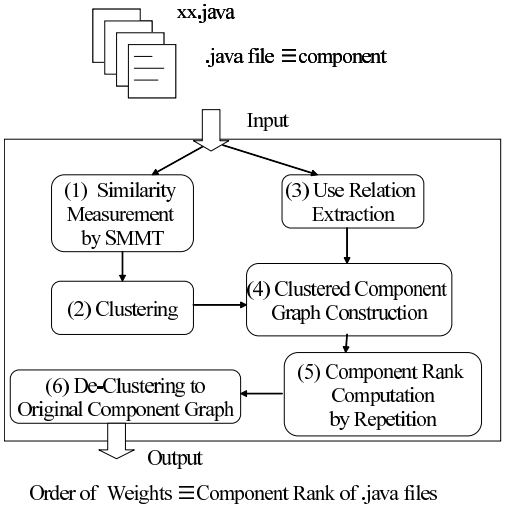


図 6　Component Rank システム（CR-システム）の構成
Fig. 6　Architecture of the Component Rank system.

［辺と擬似辺の重みの配分比率 p］ 配分比率 p として $p = 0.85$ を採用している．この場合，部品の重みの 85%が実際の辺に分配され，15%が擬似辺の重みとしてすべての部品に均等に分配される．

［分配率］ 現在は重要度計算の効率化のため，分配率を利用関係の種類にかかわらずすべて等しくしている．例えばある部品が五つの部品を利用した場合，それぞれの辺は 17%の重みをもつ．

［類似度のしきい値 t］ グループ化において部品間の類似度を評価するメトリクスとして，文献 [16] で提案されている「S_{line}」を用いる．S_{line} は二つのソースコードファイル間で一致する行の割合で，二つのファイル間の類似度を表す.「SMMT（Similarity Metrics Mesuring Tool）」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．しきい値 t については $t = 0.8$ を採用している．

3.2 CR-システムの構成

CR-システムの構成を図 6 に示す．解析の手順は以下のとおりである．

（１） 入力された M 個の部品に対して，SMMT を用いることで，各部品間の類似度を計測する．

（２） SMMT で得られた類似度をもとに，しきい値 t 以上の類似度をもつ部品をまとめることで，M 個の部品を，N 個の部品群にグループ化する．

（３） M 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 [18] を利用している．

（4） 部品群に関する情報とファイル間の関係抽出部の結果（部品間の関係）から，部品群間の利用関係を求め，部品群グラフを形成する．

（5） 部品群グラフから行列を構築し，行列における繰返し計算をもとに定常分布を求め，それを各頂点（部品）の重要度とする．

（6） 各部品が属する部品群の重要度から，各部品の重要度を求め，順位付けを行い，部品の重要度ランクを出力する．

4. 評価実験

本章では，CR-システムによって求められた部品の順位が妥当であるかを確認するために，三つのソフトウェア部品の集合に対して適用実験を行う．

4.1 JDK 1.3.0 への適用

Sun Microsystems [20] から配布されている Java 2 Software Development Kit, Standard Edition 1.3.0（以下 JDK）のソースコードを対象として評価実験を行った．JDK は 1877 ファイルで構成され，総行数は約 575000 行である．Pentium 4 2 GHz の CPU, 2 GBytes のメモリをもつ PC 上で，JDK に対して CR-システムを用いて順位付けを行ったところ，約 7 分の時間を必要とした．

表 1 は得られた順位の上位 10 位までを表にしたものである．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 [5] で利用しなければならないクラスが大半を占めている．例えば，java.lang.Class クラスは実行中のクラス及びインタフェースを表すクラスで，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．一方で最下位は 1256 位で，最下位に属する 622 クラスはどのクラスからも利用されていない．

これらの Java において重要な役割を担っているクラスが上位を占めているということから，CR 法による部品の順位付けは妥当であると考えられる．

表 1 JDK 1.3.0 における Component Rank
Table 1 Component Rank for JDK 1.3.0.

C.Rank	Class Name	Weight
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
⋮	⋮	⋮
1256	⋮	⋮

4.2 研究室内ソースコードへの適用

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（582 ファイル）を適用対象として実験を行った．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］ Java ソースコードから C-K メトリクス [3] を計測するツールとそのバージョンアップ版．パッケージ名は cktool 及び cktool_new（29+29 ファイル）．ANTLR [18] を使用してソースコードの構文解析を行っている．

［CR-システム］ 本論文で作成した CR-システム．下記で示したライブラリを利用している．パッケージ名は jp.ac.osaka_u.es.ics.iip_lab.metrics.（68 ファイル）

［ライブラリ群］ ANTLR（パッケージ名 antlr，188 ファイル），JAMA [19]（パッケージ名 Jama，9 ファイル），Caffe Cappuccino Class Library [13]（パッケージ名 jp.gr.java_conf.keisuken，245 ファイル）．

［その他］（14 ファイル）

適用結果を表 2 に示す．2 位，8 位に同じ順位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よく似ているため同一部品群に存在している

表 2 研究室内ツールにおける Component Rank
Table 2 Component Rank for SE tools.

C.Rank	Class Name	Weight
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

ことを示している．

複数のアプリケーションで共通に使用される ANTLR のようなパッケージが，研究室内で作成されたアプリケーションよりも全体的に上位にくる傾向が見られる．また，データを格納する Token，Vector，Matrix，Array クラスやデバッグ情報を出力するための Event クラスのような，ソフトウェア全体で広く利用されるクラスが高く評価されている．これらのクラスは，ソフトウェア内で参照される回数が多く，様々なクラス内で利用されていることが予想できる．ソフトウェア内での利用回数が多く，汎用的で重要な役割を担っていることを予想できるようなクラスが上位にきており，これらの結果から CR 法による順位付けは妥当であると考えられる．

4.3 部品検索への利用例

今回提案した手法の応用例として，インターネット上で公開されている大量の部品の中から，開発者の必要としている機能をもつ部品や，その機能の使い方を示している部品のような，再利用に有益な情報を提供するための部品検索システムの実現が考えられる．このとき，あらかじめすべての部品に対して CR 法を用いてその部品の重要度を 1 元的に定義し，検索結果において部品が複数検出されたときに，重要度の順に並べ換えて表示を行う．これによりよく利用される汎用性の高い部品を先頭に表示することができるため，知識の共有を円滑にでき，再利用を促進することができると考えられる．

4.2 で紹介したツール及び，テキストエディタ *JEDIT*，アプリケーションサーバ *Enhydra*，XSLT プロセッサ *saxon*，gnutella client*phex* などの SourceForge [21] で公開されている Java アプリケーション群を対象に検索の例を示す．総ファイル数は 7171 個で，これらのアプリケーションでは XML 文書に関する操作を行っている．

これらの部品を対象として，DOM ツリーの中で現れるノードの種類を得るためのメソッドである “getNodetype” について検索を行う場合を考える．今回の検索では UNIX のコマンドである *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，181 のクラスが該当し，CR-システムの順位をもとに検索結果を順位付けしたところ，表 3 のようになった．

部品全体の順位は JDK に対する適用の場合と同様に，java.lang.Object のような汎用的な部品が上位を占めるが，検索結果においてはそれらの部品はヒットせず，部品定義を行う部品とその利用方法に関する部品のみが抽出された．今回は，1 位と 2 位にそのメソッドの定義が出現し，その他のクラスは利用例であった．利用例を見ていくと，上位の利用例は DOM の解析などにおけるノードの操作（3〜5，10 位），スタイルシートなどの XML 文書の解析（6〜9 位）など一般的なものであったが，下位になればなるほど，XML で書かれた内容を解釈して実行するための FuncID クラスのような特殊な利用例が現れやすくなっていた．

表 3　検索結果（Component Rank 順）
Table 3　Search result sorted by Component Rank.

Order (C.Rank)	Class Name	Weight
1(67)	enhydra3.1..dom.Node	0.029110
2(169)	saxon7_0..saxon.om.NodeInfo	0.000969
3(275)	saxon7_0..saxon.pattern.NodeTest	0.000437
4(316)	enhydra3.1..dom.DocumentImpl	0.000368
5(355)	saxon7_0..saxon.pattern.Pattern	0.000324
6(382)	saxon7_0..saxon.Controller	0.000296
7(437)	enhydra3.1..xslt.XSLTEngineImpl	0.000241
8(446)	enhydra3.1..dom.ElementImpl	0.000235
9(500)	saxon7_0..saxon.style.StyleElement	0.000202
10(506)	saxon7_0..saxon.tree.NodeImpl	0.000198
⋮	⋮	⋮
125(4441)	enhydra3.1..FuncID	0.000029
⋮	⋮	⋮
125(4441)		

今回提案する重要度を部品検索システムにおける検索結果の表示順位決定に用いることで，今回の例のように大量の部品の中からでも部品定義に関する情報を上位に配置できる．更に，ある機能の利用方法を知りたい場合にも，一般的な利用方法から参照できるようになるため，知識の共有を円滑に行うことができる．順位付けがない場合，最悪の場合すべての部品を見る必要があることを考えると，CR 法を部品検索システムの表示順位の決定に用いることは有益であると考えられる．

5. 考　　察

5.1 関 連 研 究

5.1.1 利用回数による評価手法

本論文で提案した CR 法は，利用関係をもとに各部品の重要度を評価する手法である．各部品の重要度を評価する方法としては，各部品の利用回数をもとに評価する手法も考えられる．[11] では，クラスの被利用クラス数からクラスの人気度を測定する測定法として，

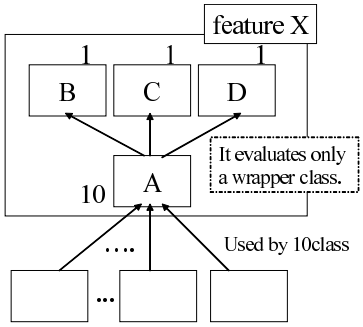


図 7　利用回数による評価における問題
Fig. 7　A problem of evaluation by the number of use classes.

CP（class popularity）が提案されている．

この手法の場合，CR 法のような繰返し計算は必要としない．しかし，今回想定しているようなソフトウェアの部品検索システムにおける利用を考えた場合，実際に機能を実現している部品を発見するのが困難になる場合があると考えられる．例えば図 7 のような，複数の部品である機能 X を実現しているような部品の集合 A，B，C，D が存在するとする．この部品の集合では，B，C，D は機能 X の一部をそれぞれ実現している一方で，A は機能 X に関して外部とのやり取りをするためのラッパークラスで，B，C，D で実現されているメソッドを呼び出すだけである．この機能 X がよく利用されている場合に評価を行う場合を考える．この場合，利用回数で評価してしまうと．検索を行ったときに橋渡しをしている A のみが評価され，実際に機能 X を実現している B，C，D が評価されない．

一方で CR 法では，A の重要度が利用関係を通じて B，C，D に反映されるため，B，C，D も比較的上位として評価することができる．実際に機能 X を実現している部品 B，C，D はただ機能 X の使い方を知りたい際にはさほど重要ではないが，機能 X を理解したり機能 X を拡張する際には極めて重要であると考えられる．再利用支援を目的とした部品検索システムの場合，前者の利用法の取得だけでなく，後者のような機能理解や機能拡張も検索の目的として考えられる．そのため，部品検索に利用する際には CR 法を用いる方法が適切であると考えられる．Java のようなオブジェクト指向言語においては，複数の部品で一つの機能を生成することが多く，そのような部品内では他のメソッドを呼び出すだけのメソッドが定義されることが多いため，今回のように Java で記述された部品に対して CR 法を適用することは適切であると思われる．

5. 1. 2　他分野における評価方法

今回提案した CR 法のような，利用関係をもとに行列を生成し繰り返し計算を行うことで評価を行う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，ある論文誌を引用する回数及び引用される回数をもとに行列を構築し，繰り返し計算を行うことで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 [10]．また，サーチエンジン Google では，検索結果の表示順を決めるために，リンク構造から行列を構築し，繰り返し計算を行うことであらゆるページの重要度（PageRank）を評価している [9], [14]．Influence Weight や PageRank と CR 法は，利用関係をグラフ及び行列に変換したのち各ノードの重みを計算しているなど，よく似ている．また，JDK の API の一部（java.lang パッケージ）の文書中からリンク構造を抽出し，PageRank 手法を適用した例 [15] では本論文における JDK への適用結果とよく似た結果が示されている．

しかし，これらの手法の対象はソフトウェア部品ではなく，参考文献，ハイパテキスト，ソフトウェア部品の内容を記述した文献など，すべて文書である．現在のところ，ソフトウェア部品に対して直接適用した事例はない．

更に，ソフトウェア部品に対してこのような手法を適用する際には，コピーの存在を考慮する必要がある．実際のソフトウェア開発現場では，既存の部品からコピーされ修正されることによってでき上がることが多く，コピーは開発期間の短縮などソフトウェアの実現に大きな意味をもつ．一方で学術論文や Web ページのコピーにはほとんど意味がない．部品においてはこのような類似部品の存在が特性となっており，評価を行う際には類似部品の存在を前提としなければならない．

インターネット上から様々なソフトウェア部品を集めた場合，収集された部品の中には，流用したためであるとか，実行に必要なライブラリを添付したためであるなどの理由で，類似部品が大量にあると考えられる．部品のグループ化を行わずにただ単に部品に記述されている利用関係を追跡するだけでは，ライブラリと各ソフトウェア間の利用関係は解析できるが，異なるソフトウェア間のコピーした，されたという関係は解析できない．この場合でも，ライブラリ中の部品の重要度は利用実績を反映していると考えられるが，ソ

フトウェア内の部品はソフトウェアごとに利用関係が独立することが考えられるため，利用実績を反映したものにならない．

ソフトウェア内の部品の利用実績を評価するためには，類似部品の存在を念頭においてどのようにその部品が生成されたかを考慮して，異なるソフトウェア間に存在するコピーに関する利用実績を定義する必要がある．ただ，コピーの関係を部品グラフ上の有向辺として定義するのは難しいため，提案手法では部品のグループ化を行うことでコピーの関係を重要度に反映させている．コピーして再利用される回数が多い（類似部品が多い）ほどその部品群を利用している部品の数は多くなるため，類似部品の重要度を高くすることができる．

5.1.3 再利用性の評価

前述のとおり CR 法は利用実績を考慮する際，部品の再利用についても考慮している．我々は，CR 法を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を定量的に評価した指標として，ソフトウェア部品の再利用性評価に利用できると考えている．

再利用は，一般に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おり，コストを削減することができた事例の報告も行われている [1], [6], [8]．これまでに個々のソフトウェア部品の再利用性を評価する方法がいくつか提案されている．例えば，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 [4]．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 [17]．

これらの手法はすべて，部品そのものから読み取れる特性のみを計算して再利用性を評価するもので，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と似ているという結果が得られている．しかし従来の手法は，部品そのものから読み取れる特性のみを評価したもので，利用実績については考慮されていない．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．CR 法は，従来の再利用性評価手法において考慮されていない部分である，利用実績に関する評価を補完するものであり，ソフトウェア部品の再利用性評価に利用できると考えている．

5.2 配分比率及びしきい値について

今回提案した CR 法では，パラメータとして辺と擬似辺の重みの配分比率 p 及びグループ化におけるしきい値 t が存在する．以下では，値の決定について考察する．

配分比率 p に関して p の値を変えて順位の変動を検証したところ，$p = 0$（補正のみしかないためすべての部品が同順位になる）と $p = 1.0$（補正なし）のとき以外は，値を変更しても結果として得られる順位に違いはほとんど見られなかった．また，$p = 1.0$（補正なし）の場合は補正を行った場合と比べて 10 倍以上の計算時間を必要とした．

このことから，p は順位の決定に影響を与えるパラメータではなく，計算の収束にのみ必要なパラメータで，p はどの値でも問題ないことがわかった．そこで，現在のところ最初に採用していた値である $p = 0.85$ を利用している．

一方で，しきい値 t に関しても t の値を変えて順位の変動を検証したところ，似ていると判定される部品が増えるため一部の部品の順位は変わるが，結果として得られる順位に大きな影響を与えるものではなかった．しかししきい値を低くした場合，判定回数が増え計算時間が膨大になるため．効率化を行うためにできるだけ高いしきい値を用いて判定対象を絞り込む必要がある．そこで，しきい値を $t = 0.8$ まであげたところ，しきい値を 0.1 にした場合と比べて時間が 1/10 以下に抑えられたにもかかわらず，しきい値 0.1 において類似と判定された部品に関してその半分以上は類似していると判定できた．

グループ化を行う際の目的が「コピーされた部品若しくは一部修正が加わった部品を検出すること」であることを考慮すると，「コピーした上で 1 割強程度の変更がなされた部品」までを把握できれば十分な精度が得られると考えられ，現在のところ $t = 0.8$ を用いている．

5.3 SPARS

私たちは CR-システムを用いたソフトウェア部品検索エンジンとして，現在 SPARS（Software Product Archiving, analyzing, and Retrieving System）を開発している．図 8 は SPARS の構成図である．

SPARS は収集してきた Java のソースプログラムに対して解析を行い，リポジトリに保存し，CR-システムを用いて各部品を順位付けする．部品を検索したい利用者はブラウザを通じて必要な部品に関する情報

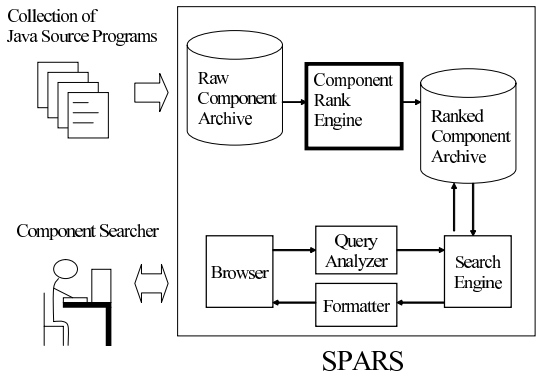


図 8　SPARS の構成
Fig. 8　Architecture of SPARS.

を検索キーとして入力する．現在のところ，キーワードによる検索を想定しているが，コード断片による検索も考慮する予定である．部品検索部は，検索キーを解析しリポジトリ内を検索し．検索にヒットした部品を CR-システムによる順位をもとに並び換えて出力する．これにより利用者はよく利用される利用実績の高い部品を容易に取得することができる．

6. むすび

本論文では，ソフトウェア部品の利用実績に基づいて各部品の重要度を測定し，順位付けし，評価する手法として CR 法を提案した．提案手法では，各ソフトウェア部品間に存在する利用関係に基づいてグラフ及び行列を構築し，構築された行列に対して繰り返し計算を行うことで，各部品の重要度を測定している．

また，実際に CR 法に基づいて部品重要度を評価する CR-システムを実装し，いくつかのソフトウェアシステムに適用した．いくつかの例に適用した結果，よく利用されるクラスや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，部品重要度は，利用実績を表すものであると同時に，ソフトウェア部品の検索のための順位付けにも有用なものであった．この CR 法を再利用する部品を選択する基準として用いることで，知識の共有をスムーズに行うことができ，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，更に多くの部品への適用，継承，実装，メソッド呼出しの各関係の重み付けの検討，部品検索システム（SPARS）の実現などが挙げられる．

謝辞　本研究は，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．

文　献

[1] V.R. Basili, G. Caldiera, F. McGarry, R. Pajerski, G. Page, and S. Waligora, “The software engineering laboratory — An operational software experience,” Proc. ICSE14, pp.370–381, 1992.
[2] G. Blom, L. Holst, and D. Sandell（著), 森　真（訳), 確率問題ゼミ，シュプリンガー・フェアラーク東京，1995.
[3] S.R. Chidamber and C.F. Kemerer, “A metrics suite for object-oriented design,” IEEE Trans. Softw. Eng., vol.20, no.6, pp.476–493, 1994.
[4] L.H. Etzkorn, W.E. Huges Jr., and C.G. Davis, “Automated reusability quality analysis of OO legacy software,” Information and Software Technology, vol.43, Issue 5, pp.295–308, 2001.
[5] J. Gosling, B. Joy, and G. Steele（著), 村上雅章（訳), Java 言語仕様，アジソン・ウェスレイ・パブリッシャーズ・ジャパン，1997.
[6] S. Isoda, “Experience report on a software reuse project: Its structure, activities, and statistical results,” Proc. ICSE14, pp.320–326, 1992.
[7] I. Jacobson, M. Griss, and P. Jonsson, Software Reuse, Addison Wesley, 1997.
[8] B. Keepence and M. Mannion, “Using patterns to model variability in product families,” IEEE Software, vol.16, no.4, pp.102–108, 1999.
[9] L. Page, S. Brin, R. Motwani, and T. Winograd, “The PageRank citation ranking: Bringing order to the web,” http://www-db.stanford.edu/~backrub/pageranksub.ps
[10] G. Pinski and F. Narin, “Citation influence for journal aggregates of scientific publications: Theory, with application to the literature of physics,” Information Processing and Management, vol.12, no.5, pp.297–312, 1976.
[11] 青木　淳，オブジェクト指向システム分析設計入門，ソフトリサーチセンター，1992.
[12] 青山幹雄，中所武司，向山　博，コンポーネントウェア，共立出版，1998.
[13] 西本圭佑，“Java 言語について，” http://cappuccino.ne.jp/keisuken/java/
[14] 馬場　肇，“Google の秘密 — PageRank 徹底解説，” http:// www.kusastro.kyoto-u.ac.jp/~baba/wais/pagerank.html
[15] 山本　篤，“Google の基礎技術 PageRank について，” KTYY Seminar, 2001, http://aglaia.c.u-tokyo.ac.jp/~yamamoto/KTYY_Seminar/20010220/
[16] 山本哲男，松下　誠，神谷年洋，井上克郎，“ソフトウェアシステムの類似度とその計測ツール SMMT，” 信学論（D-I), vol.J85-D-I, no.6, pp.503–511, June 2002.
[17] 山本浩数，鷲崎弘宜，深澤良彰，“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.
[18] “ANTLR Website,” http://www.antlr.org/
[19] “JAMA: A Java Matrix Package,” http://math.nist.gov/javanumerics/
[20] “The Source For Java(TM) Technology,” Sun Microsystems, http://java.sun.com/
[21] “SourceForge,” http://sourceforge.net/

（平成 14 年 10 月 21 日受付，3 月 28 日再受付）

横森　励士　（学生員）

平 11 阪大・基礎工・情報卒．平 13 同大大学院修士課程了．現在，同大学院博士課程在学中．プログラム構造解析の研究に従事．

藤原　　晃

平 14 阪大大学院修士課程了．現在，松下電器産業（株）に勤務．在学中，ソフトウェアの生産性や品質の定量的評価に関する研究に従事．

山本　哲男

平 9 阪大・基礎工・情報卒．平 11 同大大学院博士前期了．平 14 同大学院博士後期課程了．現在，科学技術振興事業団計算科学技術研究員．博士（工学）．ソフトウェアプロセス，ソフトウェア開発支援の研究に従事．

松下　　誠

平 5 阪大・基礎工・情報卒．平 10 同大大学院博士後期課程了．同年同大・基礎工・情報・助手．平 14 阪大大学院情報科学研究科・助手．工博．ソフトウェアプロセス，オープンソースソフトウェア開発の研究に従事．

楠本　真二　（正員）

昭 63 阪大・基礎工・情報卒．平 3 同大大学院博士課程中退．同年同大・基礎工・情報・助手．平 8 同大講師．平 11 同大助教授．平 14 阪大・情報・コンピュータサイエンス・助教授．博士（工学）．ソフトウェアの生産性や品質の定量的評価，プロジェクト管理に関する研究に従事．情報処理学会，IEEE，IFPUG 各会員．

井上　克郎　（正員）

昭 54 阪大・基礎工・情報卒．昭 59 同大大学院博士課程了．同年同大・基礎工・情報・助手．昭 59〜61 ハワイ大マノア校・情報工学科・助教授．平元阪大・基礎工・情報・講師．平 3 同学科・助教授．平 7 同学科・教授．平 14 阪大大学院情報科学研究科・教授．ソフトウェア工学の研究に従事．情報処理学会，日本ソフトウェア科学会，IEEE，ACM 各会員．