社団法人　電子情報通信学会
THE INSTITUTE OF ELECTRONICS,
INFORMATION AND COMMUNICATION ENGINEERS

信学技報
TECHNICAL REPORT OF IEICE.

利用頻度に基づくソフトウェア部品の解析・検索システムの提案

山本　哲男† 　横森　励士†† 　松下　誠††† 　楠本　真二††† 　井上　克郎†††

† 科学技術振興事業団　〒 332-0012 埼玉県川口市本町 4–1–8
†† 大阪大学大学院基礎工学研究科　〒 560–8531 大阪府豊中市待兼山町 1–3
††† 大阪大学大学院情報科学研究科　〒 560–8531 大阪府豊中市待兼山町 1–3
E-mail: †{t-yamamt,matusita,kusumoto,inoue}@ist.osaka-u.ac.jp, ††yokomori@ics.es.osaka-u.ac.jp

あらまし　再利用性の高いソフトウェア部品を再利用することにより，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，ほとんどは，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った．しかし，再利用性が高いということは，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よって定量的に示されることが必要である．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解析・検索システム SPARS(Software Product Archive, analysis and Retrieval System) の提案を行う．SPARS は Java を対象としたシステムであり，キーワードやコード片等を検索キーとして，検索キーと関連したソフトウェアのソースコード等を容易に検索できるシステムである．

キーワード　ソフトウェア部品，再利用，Java

Software Component Analysis and Retrieval System Based on Reusability

Tetsuo YAMAMOTO†, Reishi YOKOMORI††, Makoto MATSUSHITA†††, Shinji KUSUMOTO†††, and Katsuro INOUE†††

† Japan Science and Technology Corporation
4–1–8, Honmachi, Kawaguchi-shi, Saitama 332–8531, Japan
†† Graduate School of Engineering Science, Osaka University
1–3, Machikaneyama-cho, Toyonaka-shi, Osaka 560–8531, Japan
††† Graduate School of Information Science and Technology, Osaka University
1–3, Machikaneyama-cho, Toyonaka-shi, Osaka 560–8531, Japan
E-mail: †{t-yamamt,matusita,kusumoto,inoue}@ist.osaka-u.ac.jp, ††yokomori@ics.es.osaka-u.ac.jp

Abstract　Reusing the software components that have high reusability improve the software productivity, quality and cost. A lot of reusability metrics have been proposed. Most of them are based on the calculation of the static attributes of the software component. However, in practice, it is necessary to show quantitatively that the components with high reusability have actually reused in many software systems. There may exist such software component that have low reusability based on the conventional metrics but actually have reused in many software systems. This paper proposes the new reusability measurement method. Base on the proposed method, we have developed SPARS(Software Product Archive, analysis and Retrieval System).

Key words　Software component, Reuse, Java

1. はじめに

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た．これを実現するために様々なソフトウェア工学技術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 [4]．一般にソフトウェアの再利用は生産性と品質を改善し，結果としてコスト削減する報告が多く出されている [3] [7] [9]．

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 [6]．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 [14]．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，再利用性が高いということは，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よって定量的に示されることが必要である．仮に，主観的な判断で再利用性が高いと判断されたとしても，実際の利用実績がなければその判断には意味がないであろう．実際には，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．

我々はこれまでに，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してきてる [15]．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（再利用性が高い）．

このような評価手法は，様々な分野において採用されている．論文の引用解析の分野においては，1970 年代には実績に基づいて論文の重要度を評価する手法が提案されている．Narin らは，論文の重要度を論文がどの程度引用されているかという実績に基づいて評価する手法「Influence Weight」を提案している [10], [12], [13]．この手法では，(1) 多くの論文から引用されている論文は重要である，(2) 重要な論文から引用されている論文はまた重要である，という 2 つの考えに基づいて論文の重要度を評価している．また，よく知られている Web ページサーチエンジン 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を評価している [2], [11]．

この考え方に基づいて，評価手法を定義した [15]．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．利用関係はそれぞれ再利用性評価値の重みを持つ．部品の再利用性評価値は，その部品が利用される関係の重みの合計値となる．部品の再利用性評価値を求める計算は，行列の固有値計算に帰着される．

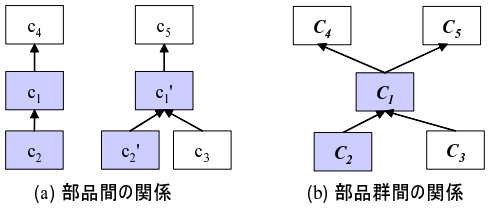

図 1　類似した部品群

本論文では，提案した手法に基づいたソフトウェア部品の解析・検索システムである，SPARS(Software Product Archive, analysis and Retrieval System) を提案する．SPARS は，ソフトウェアの容易な検索，参照を実現するためのシステムである．キーワードやコード片等を検索キーとして，検索キーと関連したソフトウェアのソースコード等を効率よく検索できることを目標としている．

以降，2. 節では，本研究における相対的再利用性とその評価手法について述べる．3. 節では，提案するソフトウェア部品の解析・検索システム SPARS について説明する．4. 節では，適用実験について述べる．

2. 準　　備

本節では，文献 [15] で提案したソフトウェアの相対的再利用性評価手法について簡単に説明する．

2.1 ソフトウェア部品

一般にソフトウェア部品 (Software Component) 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], [8]．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．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，コピーして一部変更した部品が多く存在する．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．

ある部品群 C_i に属する部品が，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．まず，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 $m(0 \leqq m \leqq n)$ 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 $t(0 \leqq t \leqq 1)$ 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 [6].

また，山本らはソースコードが非開示な部品に対して，そのインタフェー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 [14].

これらの方法は全て，部品の構造やインタフェー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．

2.4 相対的再利用性評価手法 R^3 法

開発者が部品を利用するということは，その部品に対して「再利用性が高い」という支持投票をしたとみなす．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，支持投票数が増加していく．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．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．

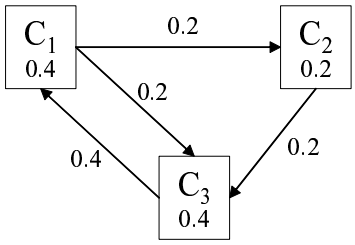

図 2　再利用性評価の例

また，個々の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には，各利用部品が A の機能に占める割合が少なくなるので，部品 A の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)

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求められるため，部品の集合の要素数や利用関係が変化すると，変化前の評価値と変化後の評価値では比較することができない．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．

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 [15].

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 となるようにしておく．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. SPARS システム

本節では R^3 法に基づいてソフトウェア部品の解析，検索を行う SPARS のシステム構成について説明する．まず，システムアーキテクチャの概要を説明した後，個々の構成部品について詳しく述べる．

3.1 システム概要

システムのアーキテクチャを図 3 に示す．システムが対象と

するソフトウェア部品は Java のクラスとする．Java で書かれたソースファイルを Raw Source Repository に格納する．新規に格納された Java ファイルは，Parser に渡され，ファイルの構文解析が行われる．構文解析された結果は検索を行うために必要なため，Source Repository に格納される．また，同時に特徴メトリクスを計測し，部品（クラス）間利用関係の抽出も行う．抽出した利用関係は Relation Repository に格納する．構文解析されたファイル中の類似ファイル群の検索は Analyzer によって行われる．計測したメトリクスから類似した Java ファイルを検索し，一つのファイル群にまとめる．ファイル群にまとめられた Java ファイルの利用関係もまとめて，Relation Repository に格納する．

Ranker は Relation Repository を参照して，Java ファイルの利用頻度を計算する．それらの値は Raw Source Repository の Java ファイルに対して関連付けされる．

利用者によって与えられた検索キーは Query Parser に渡され，検索の種類を判別し，実際に検索を行う部分である Searcher が検索を行う．Searcher は Source Repository から検索キーに一致する部分を見つけだす．一致した部分が存在すれば Browser により，一致した部分と，その Java ファイルについての情報を利用者に表示する．

3.2 Repositry と Parser

Raw Source Repository

Raw Source Repository は Java ファイルをそのままの形で保存しておく貯蔵庫である．解析対象である Java ファイルは，まず最初に Raw Source Repository に格納する．格納する際に，パッケージ名とクラス名を元に既に同じファイルが存在するかどうか調べる．もし，存在しなければ，新規ファイルとして格納し，一意な ID 番号を与える．存在していれば，ファイルの更新とみなし，Raw Source Repository のファイルを新しいファイルに置き換える．

Parser

Raw Source Repository に Java のファイルを追加すると，次に，そのファイルを Parser で解析する．Parser は Java ファイルの構文解析を行う．構文解析した後，部品の利用関係の抽出を行う．本システムで利用関係として抽出するものを以下に示す．

- クラスの継承関係
  子クラスが親クラスを利用している．
- インタフェースの実装関係
  実装クラスがインタフェースを利用している．
- 抽象クラスの実装関係
  実装クラスが抽象クラスを利用している．
- メソドの呼び出し関係
  呼び出し元のクラスがメソド呼び出し先のクラスを利用している．
- フィールドの参照関係
  参照元のクラスがフィールドの参照先のクラスを利用している．

上記の内，メソドの呼び出し関係とフィールドの参照関係は動的に決定する．そのため，構文解析だけでは，これらの関係を一意に特定する事は不可能である．一意に特定するためには，Java の実行が必要不可欠であり，すべての Java ファイルに対して実行を行うのは多くのコストがかかる．また，実行する際に入力が必要な場合もあり，自動化することが難しい．そのため，呼び出し先，または参照先のクラスを静的解析を行い，宣言されているクラスに対して利用関係を抽出する．

また，抽出したそれぞれの利用関係に対して重みを与えるかどうかが考えられる．継承，実装関係と呼び出し，参照関係で異なる重みを与えることで，利用頻度を計算した結果が変わる可能性がある．どの利用関係にどの程度の重みを与えると，良い結果が得られるかは知られていない．

Parser では，利用関係の抽出の他に特徴メトリクスの計測を行う．Java ファイルの特徴メトリクスを計測し比較することで，類似ファイルの検索が行える．類似ファイルを検索することで部品群を作成することが可能になる．実際に，類似ファイルを検索する部分は Analyzer であり，Parser では，特徴メトリクスの計測だけを行う．計測するメトリクスのいくつかを以下に示す．

- クラス名
- フィールド名
- メソド名
- メソドのシグネチャ
- メソドの複雑度

すべてのメトリクスは構文解析時に計測可能である．

Source Repository

Parser によって解析した情報は Source Repository へ格納する．構文解析した結果と特徴メトリクスの値を ID 番号と関連付けし，格納する．また，Source Repository は検索の際にも使用するため，各解析結果と ID 番号の逆引きの表を作成する．

Relation Repository

各 Java ファイル間の利用関係を格納する Repository である．利用関係は，利用関係の種類と関係を結ぶ ID 番号の組から構成される．これらの情報を格納する．逆に，任意の ID 番号を選んだ際に，どの ID 番号とどのような関係が存在するかを高速に検索するために，ID 番号の数だけ逆引きの表を作成する．

3.3 Analyzer

Parser で求めた特徴メトリクスを用いて類似ファイルを検索する．類似ファイルと判断されたファイル同士を同一部品群とする．この際，各ファイルの利用関係を一つにまとめる必要がある．そのため，Relation Repository から各ファイルの利用関係を検索し，部品群の利用関係として格納し直す．また，部品群がどのファイルから構成されているかの情報は Source Repository 格納する．

ファイルが類似しているかどうかの判別は特徴メトリクスの一致している割合から判断する．一致しているかどうかの割合を変化させることで，部品群を構成するファイルが変化する．

3.4 Ranker

Relation Repository に格納された部品群間の利用関係から相対的利用頻度の計算を行う．最初に，Relation Repository に

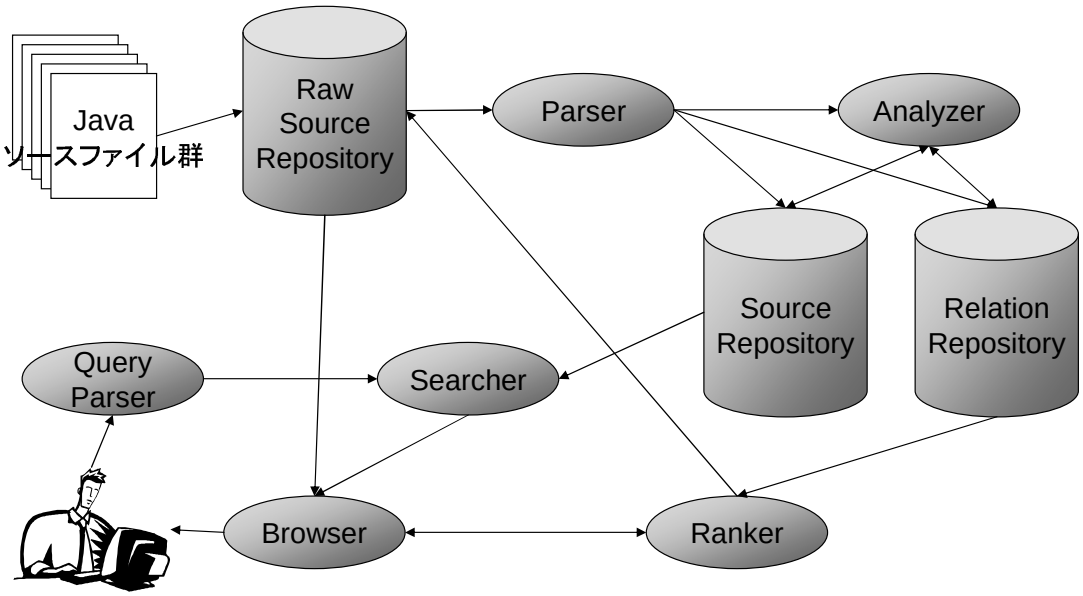

図 3　SPARS アーキテクチャ

格納されている利用関係を取り出す．部品の再利用性評価値を求める計算は，行列の固有値計算に帰着される．固有値計算を行い，利用頻度の順位を決定した後，その結果を ID 番号と組にして Raw Source Repository に格納する．

3.5　Searcher

Searcher は検索キーを元に Source Repository から必要なファイルを検索する．システムの利用者は Query Parser を通じて検索キーを入力する．入力可能なキーの種類は，キーワードとコード片である．また，検索対象は，コメントとソースコードの文である．検索の種類に応じて Searcher は Source Repository の適切な部分のみを検索し，キーと一致した部分とその部分を含むファイルの ID 番号を作成する．すべての検索が終了すると，その検索結果を Ranker で求めた順位順に並び替えを行い，Browser で表示する．

Browser は検索キーが一致したファイルの箇所とそのファイルを表示する．検索を行った結果，利用者に必要なファイルは Browser から取得可能である．

4. 適 用 実 験

提案するシステムのプロトタイプを作成し，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．適用対象は，Java 2 Software Development Kit, Standard Edition 1.3.0 (以下 JDK) および 10 個のアプリケーションの Java のソースコードで，部品数は 6426 ファイルである．実験においては，相対的再利用性評価値によって求めた順位と，各部品群の被利用回数による順位を比較することで，計算手法の有効性の評価を行う．また，部品群化を行った場合と，そうでない場合の順位の比較を行う事で，部品群化手法の有効性の評価を行う．

4.1　計算手法の有効性の評価

表 1，2 は，それぞれ相対的再利用性評価値によって求めた順位と，各部品群の被利用回数による順位である．

両者を比較してみると，各部品群の被利用関係のみによる順位付けの場合，よく利用されているという観点のみで評価されており，システムの末端においてよく利用される部品が上位に来やすいという傾向が現れている．それに対して，相対的再利用性評価値における順位ではよく利用されているという観点に加えて，部品群全体の中での重要度を加味している傾向が現れた．例えば，データ構造を定義したクラスなどは，システム内において重要な位置を占めるため，被利用関係のみの場合と比較して上位に現れた．

相対的再利用性評価値によって求めた順位と，各部品群の被利用回数による順位を spearman の順位相関係数で比較したところ，0.992 と非常に高い値が現れた．理由としては，解析対象が比較的小さかったことと，解析対象内のアプリケーションにおける部品において質の低い部品があまり存在しなかったことがあげられる．ただ実利用を考えた場合，解析対象が大きくなることで質の低い部品が増加し，質の低い部品によって利用回数が歪められてしまい，被利用回数による順位付けがうまくいかない可能性がある．一方で，相対的再利用性評価値による順位付けでは利用関係が利用する側の評価値に基づいて重み付けされるため，質の低い部品の影響を受けにくく，実利用を考えた場合有効な手法であると思われる．

4.2　部品群化の有効性の評価

表 3 は，部品群化を行なわなかった場合の相対的再利用性評価値による順位である．

部品群化によって順位が上昇した部品のほとんどが，部品群が複数のシステムにまたがったため，部品群間の辺がまとまらず利用関係が増えたため評価が高くなった部品および，評価が高い部品とよく似た部品であったため評価が高くなった部品だった．特に大きく現れたのは，例外を扱うクラスで，例外処

表 1　相対的再利用性評価値による順位

順位	クラス名	評価値
1	java.lang.Object.java	0.130842077
2	java.lang.Class.java	0.070779904
3	java.io.EOFException.java	0.051143644
≈	...	≈
3	java.io.IOException.java	0.051143644
66	org.w3c.dom.Node.java	0.034599252
67	java.lang.Throwable.java	0.03047256
68	java.lang.StringBuffer.java	0.014695051
69	javax.naming.NamingException.java	0.013914379
70	java.lang.VirtualMachineError.java	0.010906467
71	org.w3c.dom.NodeList.java	0.010547134
72	java.io.InputStream.java	0.010030122

表 2　被利用回数による順位

順位	クラス名	評価値
1	java.lang.Object.java	1201
2	java.io.EOFException.java	1120
≈	...	≈
2	java.io.IOException.java	1120
65	java.util.Vector.java	545
66	java.lang.StringBuffer.java	458
67	java.util.Hashtable.java	448
68	java.util.HashMap.java	448
69	java.util.Enumeration.java	444
70	java.io.File.java	326
71	java.lang.Class.java	297
72	org.w3c.dom.Node.java	285
73	java.awt.Component.java	263

表 3　部品群化を行わなかった場合の順位

順位	クラス名	評価値
1	java.lang.Object.java	0.137758265
2	java.lang.Class.java	0.074010154
3	java.lang.Throwable.java	0.051009801
4	org.w3c.dom.Node.java	0.029174933
5	java.lang.Exception.java	0.028564972
6	java.lang.StringBuffer.java	0.015180348
7	java.io.IOException.java	0.012814825
8	java.lang.SecurityManager.java	0.00959242
9	java.io.InputStream.java	0.00916187
10	org.w3c.dom.NodeList.java	0.009015037

理が非常によく再利用されている事を示している．

一方で，再利用が行われずに単一システム内で部品群が閉じてしまっている場合，辺の本数が増えず評価値は上がらないケースがよく現れた．これは単一システム間のなかでは，似た利用方法になりやすいため，利用側も被利用側もどちらもそれぞれ一つの部品群に部品群化されやすく，部品群化時に利用関係がまとめられてしまう事が原因である．

部品群化によって，高い評価の部品とほぼ同一の部品に対しては同一のランク値を与えることができ，システム間にまたがるような利用関係が多い部品は評価値が高くなることを示す事ができているため，コピーされた部品の把握およびシステム間の横のつながりを表現出来ているといえ，部品群を行うことは有効であることがわかる．

5. ま と め

本論文では，相対的再利用性に基づくソフトウェア部品の解析・検索システム SPARS を提案した．さらに，提案システムが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．

今後の課題としては，SPARS の実装，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] 青山, 中所, 向山: コンポーネントウェア, 共立出版, (1998).
[2] 馬場: “Google の秘密 - PageRank 徹底解説”, http://www.kusastro.kyoto-u.ac.jp/ baba/wais/pagerank.htm
[3] 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).
[4] C. Braun: Reuse, in John J. Marciniak, editor, *Encyclopedia of Software Engineering*, Vol. 2, John Wiley & Sons, pp. 1055-1069 (1994).
[5] 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).
[6] 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).
[7] S. Isoda: “Experience report on a software reuse project: Its structure, activities, and statistical results”, *Proc. of ICSE14*, pp.320-326 (1992).
[8] I. Jacobson, M. Griss and P. Jonsson: Software Reuse, *Addison Wesley*, (1997).
[9] B. Keepence and M. Mannion: “Using patterns to model variability in product families”, *IEEE Software*, Vol. 16, No. 4, pp. 102-108 (1999).
[10] F. Narin, G. Pinski, and H. H. Gee: “Structure of the Biomedical Literature,” *Journal of the American Society for Information Science*, Vol. 27, No. 1, 25-45, (1976).
[11] 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
[12] 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) .
[13] G. Pinski and F. Narin: “Structure of the Psychological Literature,” *Journal of the American Society for Information Science*, Vol. 30, No. 3, pp. 161-168, (1979).
[14] 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).
[15] 横森, 藤原, 山本, 松下, 楠本, 井上: “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”, ソフトウェア・シンポジウム 2002 論文集, pp.216–225, July 16–19, (2002).