修士学位論文

題目

Java ソフトウェア部品検索システム SPARS-J の実験的評価

指導教官
井上克郎 教授

報告者
梅森 文彰

平成 16 年 2 月 12 日

大阪大学 大学院情報科学研究科
コンピュータサイエンス専攻 ソフトウェア工学講座

平成 15 年度 修士学位論文

Java ソフトウェア部品検索システム SPARS-J の実験的評価

梅森 文彰

内容梗概

ソフトウェアの大規模化と複雑化に伴い，ソースコードなどのソフトウェアの構成要素が，再利用やソフトウェア理解などの二次利用のためにソフトウェア部品として蓄積されるようになった．しかし，効果的に再利用するためには，個々の部品の関連性を把握した上で，適切に部品を選択することが要求される．我々の研究グループでは，Java を対象としたソフトウェア部品検索システム SPARS-J に関する研究を行っている．SPARS-J では，利用関係や類似部品などのソフトウェアの特性を利用することで，部品に対する順位付けを行い，検索結果を提供する．SPARS-J を用いることで，ソフトウェア部品の知識が無い開発者も有用なソフトウェア部品やそれに付随する有益な情報を平易に入手することが期待できるが，具体的な評価は行われておらず有効性の検証が必要である．

本研究では，SPARS-J に対して実験的評価を行い，本システムの有効性を検証する．実験においては，(1) 順位付けの性能について一般的な検索システムとの比較，(2)SPARS-J 内で利用される各順位付け手法の比較，(3) 企業内のソフトウェアに対する SPARS-J の適用，を行った．実験の結果，(1) 一般的な検索システムと比較して，SPARS-J の順位付けは優れていること，(2) ２つの順位付け手法にはそれぞれ特徴があり，それらを統合した順位付け手法が一番よい結果を示すこと，(3) アンケートの結果から，SPARS-J が蓄積された部品の管理に極めて有効であること，がわかった．

以上のことから，SPARS-J は検索機能を利用した再利用に加えて，ソフトウェア理解などを目的とした部品の管理にも有効に利用できることがわかった．SPARS-J を用いることで，実際のソフトウェア開発におけるさらなる効率化が期待できる．

主な用語

ソフトウェア検索
ソフトウェア再利用
Java
実験的評価

目 次

1 まえがき

近年のソフトウェアの大規模化と複雑化に伴い，開発されたソフトウェアの有効な二次利用が，実際の開発現場において極めて重要になりつつある．ここで言うソフトウェアの二次利用とは，開発されたソフトウェアに関する (商業) 活動や開発されたソフトウェアの稼動と対比させたもので，ソフトウェアの再利用や生成されたソフトウェアの管理など，開発作業の効率化を目指す試みを指す．

既存のソフトウェアの有効的な活用方法として，ソフトウェアの部品化があげられる．ソフトウェアの部品化を行い，個々の部品の関連性を把握することで，適切な部品を組み合わせてソフトウェアを開発 (ソフトウェア部品の再利用) したり，保守作業においてソフトウェアを把握 (ソフトウェア理解) したりすることが容易となる．実際にオブジェクト指向設計開発はソフトウェアを部品化する手法として注目を集めており，再利用に関する研究も盛んである [7],[15]．

我々の研究グループでは，Java ソースコードの集合を対象としたソフトウェア部品検索システム **SPARS-J**(*Software Product Archive, analysis and Retrieval System for java*) に関する研究 [23],[31] を行っている．SPARS-J では，ソースコード内のクラスをソフトウェア部品とみなして，大量のソフトウェアから自動的に部品を抽出する．そして，それぞれの部品間の利用関係や類似といったソースコード固有の特性を考慮して，順位付けを行う．さらに，検索結果表示の際にソフトウェア部品に関する詳細な情報を併せて提供する．このシステムを用いることで，ソフトウェア部品の知識が無い開発者も有用なソフトウェア部品やそれに付随する有益な情報を平易に入手できる．しかし，SPARS-J のソフトウェア部品検索システムとしての有効性についての評価は行われておらず，システムの有効性の評価が必要である．

情報検索システムにおける評価とは，一般に順位付けの性能が評価される．また，そのときに用いられる尺度としては**適合率**(*precision ratio*) と**再現率**(*recall ratio*) が有名である．しかし，再現率を求めるためにはデータベース中の全適合文書数を把握する必要があるため，実験的環境でしか算出できず，また，大規模なデータベースにおいては，各検索要求について適合する文書を網羅的に調べるには多くの労力が必要であり，非常に困難である．また，安易な方法で適合文書の集合を作成すると，再現率が不当に高く評価されるという危険もある．このような危険を回避し，正当な評価を行うためにも，標準的テストコレクション [25] は，情報検索研究の基盤として重要である．文書検索システムにおいては，既に BMIR-J2[5] 等の利用可能なテストコレクションが存在しており，各文書検索システムの比較・評価を容易に行うことが可能である．しかし，ソフトウェア部品検索システムを対象としたテストコレクションは現在のところ存在しておらず，再現率を求める評価実験は非常に困難である．

一方で，現在の情報検索システムでは，利用者が入力した問い合わせに適合している順に検索結果が表示される「**適合度順出力**」が主流である．このような検索システムにおいては，検索結果全体の再現率よりも上位に最も適合している部品が来ることが重要であると考えられる．なぜなら，ユーザは検索結果においてその文書の多少に関わらず全ての部品を閲覧するようなことはしない．最初のページに適合文書が見つからなかった場合には，次ページを閲覧するよりは検索キーワードを変えるという傾向がある [2]．そのため，検索結果における順位は極めて重要で，またユーザの想定している順位との差が小さいほど，ユーザの意図に沿った検索を行うことができると考えられる．

本研究では，SPARS-J について，ソフトウェア部品検索システムとしての実験的評価を行う．本実験では，再現率の代わりに順位の差を表す指標をを用いて評価実験を行う．用いた尺度は文献 [32] に述べられている ndpm 値で，ndpm 値は理想的な順位と，得られた順位との距離を評価するための尺度である．実験においては，オープンソースから構成されるシステムと，企業内で開発されたソフトウェアからなるシステムを構築し，それぞれにおいて適合率と ndpm 値についての評価を行う．

オープンソースから構成されるシステムに関する評価では，多くの検索目的で用いられることの多い Web ページ検索エンジンである Google[11] とフリーウェアでありながら官公庁，市役所，企業，大学など幅広く導入実績から信頼性の高いシステムである Namazu[22] との比較対象実験を行うと共に，SPARS-J で用いられている 2 種類の順位付け手法の評価を行った．Google との比較においては，データベースが SPARS-J のデータベースとは異なっているものの，Google 自体のデータベースが非常に大規模なものであるので十分比較可能であると考えられ，また Namazu においてはデータベースを全く同じものにして比較を行うことができる．

また，企業内で開発されたソフトウェアからなるシステムにおける評価では，企業側から提供される情報が制限されているため，Google および Namazu との比較を行うことはできないため，SPARS-J 内の各順位付け手法の比較のみを行う．また，被験者に対してアンケートを実施しその評価についても考察する．

以降，2 節で SPARS-J について簡単に説明する．3 節では，評価実験を行いその結果と分析について述べ，更にアンケートによる評価も行う．4 節で実験的評価を通しての考察を行い，5 節でまとめと今後の課題について述べる．

2 SPARS-J

2.1 概要

SPARS-J(*Software Product Archive, analysis and Retrieval System for Java*) は，Java 言語 [16] で記述されたプログラムを対象としたソフトウェア部品検索システムである．Java ソースコード中のクラスやインタフェースをそれぞれ一つの部品として扱う．Java はオブジェクト指向に基づいた言語であり，クラス・インタフェースはソフトウェア部品の概念を満たす．SPARS-J はプログラム理解，保守といったより広範囲な利用目的を想定している．そのため，部品の詳細を利用者が把握可能なソースコードという形態が適切だと考えた．その他ソースコードの配布により，利用者の目的に応じたカスタマイズが可能であるという利点が存在する．同時に，単に部品を再利用したいという場合にはクラスや Java に関する知識が必要となるが，解析された部品の様々な詳細情報を提供することで，その欠点も補うことができると考えている．

SPARS-J は，Java の利用者が指定したキーワードと関連する部品を検索する．検索結果はキーワードの出現頻度と，我々が過去提案した利用実績に基づくソフトウェア部品重要度評価手法によって順位付けされる．さらに，検索結果に併せて部品間の利用関係や類似部品に関する情報を提供することで，再利用やプログラム理解・保守を行なうことが可能となる．

図 1 に SPARS-J システムの利用時のイメージを示す．本システムの特徴はサーバで稼働させるので，クライアントにシステムをインストールする必要がなく，Web ブラウザさえあればシステムが利用可能な点である．

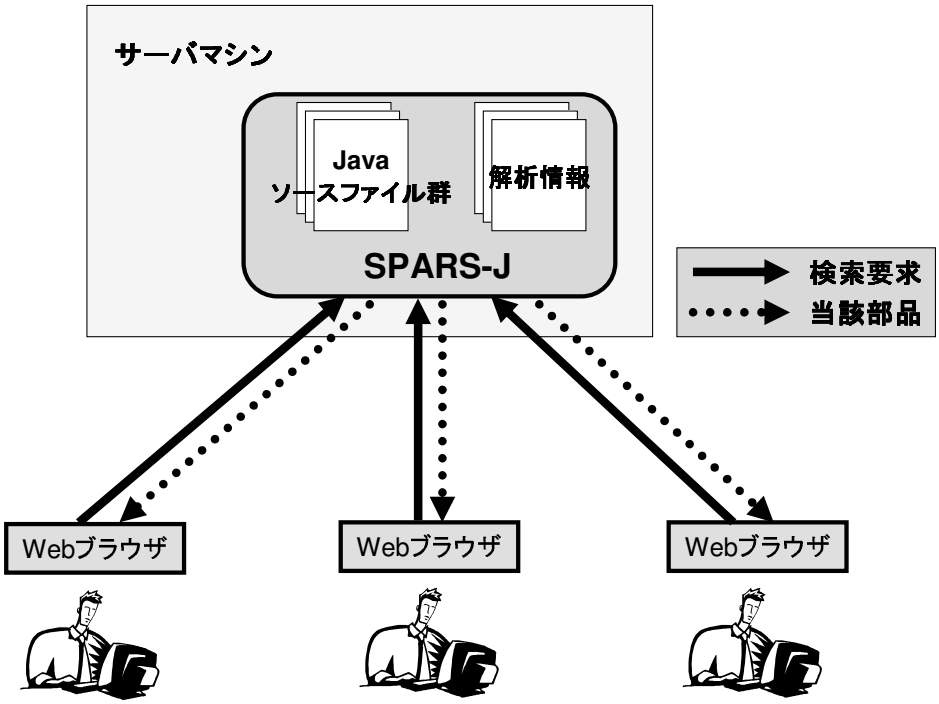

図 1: SPARS-J システムの利用イメージ

2.2 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**(*reuse*) できるように設計されたソフトウェアの実体とされ [8],[19]，過去のソフトウェア開発における成果物などからなるソフトウェア部品の集合を**ライブラリ**(*library*) という．以下，ソフトウェア部品を単に部品と呼ぶ．ライブラリ中の既存の部品を同一システム内や他のシステムで用いることを再利用といい [7]，再利用はソフトウェア生産性と品質を改善し，結果としてコスト削減するという報告が多く出されている [4],[14]．部品という概念はプログラムソースコードや実行形式ファイル，その他に設計仕様や構造，テスト項目，そしてそれらの文書などであったりと，ソフトウェア開発過程で生成されたあらゆる成果物に適用され，その種類は様々である．部品の再利用可能な度合を**再利用性**(*reusability*) といい，特定の設計やコーディングの標準に従うことで，部品の再利用性を向上させることができる．

2.3 検索システム

ソフトウェア部品検索システムとは，部品を用途や信頼度に応じて分類してライブラリとして蓄積し，ユーザの指定に応じて部品を探し出す自動化システムである．支援対象は，設計者やプログラマなど部品を検索し使用するソフトウェア開発者と，ライブラリの部品を分類し維持管理にも責任を持つライブラリ管理者である．

ソフトウェア部品検索システムが果たす役割は，主に以下の三点である．

1. 部品の分類
   部品の分類は，ソフトウェア部品検索システムにとって重要な課題である．ネジのような機械部品には規格や用途に関する違いが明確に存在しているためその分類は容易だが，ソフトウェア部品には明確な規定や分類の指針が存在しない．整理の良い開発者であれば自身が過去に開発した部品を探しだして使うことはあるかもしれないが，他人の開発した部品から用途に適合するものを探し出すことは難しい．そのため，ライブラリは適切な分類手法を用いて適切に整理されている必要がある．現在，WWWを通じて大量の部品が比較的容易に入手可能であるが，それらは自動的に分類されることが望ましい．

2. 部品の検索
   部品の検索は，分類されたライブラリから必要な部品を探し出す作業である．自然言語文書の検索システムでは検索しようとするトピックのキーワードをクエリに用いて検索するのが普通であるが，ソフトウェア部品検索システムではその他に部品の特徴をあらわすメトリクスや，求める機能，記述されている言語などをクエリに用いる場合もある．

3. 部品の収集
   部品の収集は，近年のWWWの発展に伴い注目されてきた．部品の入手先が企業の閉じたネットワーク内に限られていた時代には，先述した部品の分類と検索に関する問題が焦点であった．しかし，WWWの発展によって大量の部品を入手可能になると，WWW上のソフトウェア部品をライブラリと見なしてどのように部品を収集するかといった問題に関する手法が多く登場している．

オープンソースソフトウェア開発コミュニティSourceForge[29]では，ソースコードのストレージやコミュニケーションに必要なメーリングリストなどソフトウェア開発に必要とされる物的資源を提供することを主な目的としている．保持するソフトウェアプロダクトが増加するにつれて，開発基盤としての価値だけでなくソフトウェア部品を検索，発見する場としての価値も高まっている．例えばSourceForgeは2004年2月現在75,000以上ものプロ

ジェクトを保持しており，開発者として登録されているユーザ数は 78 万を越える．WWW 上には他にもソフトウェア開発コミュニティも存在しており，以後も発展していくとみられている．このような事実から，現在，大規模なソフトウェア部品集合に対するソフトウェア部品検索システムが求められている．

2.4 順位付け手法

部品検索をキーワードによって行うと，検索キーと合致する索引キーを持つものが多数存在する場合がある．そのため，検索結果を提示する際に適当な順位で表示することが必要となる．自然言語文書検索システムで一般的に用いられている手法では，検索対象となる文書集合から各文書の特徴を表すキーワードである索引語を抽出し，索引語の集合によってその文書の内容を近似する．登録するそれぞれの文書の特徴を的確に表すように付与された，索引語の集合などからなる情報のことを索引キーと呼ぶ．検索キーは検索者の要求の内容を近似しているため，検索キーと索引キーを用いて検索キーと文書の適合度を測り，順位付けするのが一般的である．

しかし，ソフトウェア部品を対象としてを検索する場合は，検索者の要求の内容と適合する部品であるかどうかの他，その部品がよく利用されている部品かどうかに関しても考慮する必要がある．検索キーと適合度が高い部品よりも，よく利用されている部品を上位に提示した方が，利用例も多く，部品の再利用をスムーズに行ない易いと考えられる．そのため，利用しやすい部品かどうか定量的に評価するための指標を導入し，検索キーと部品の適合度も併せて両面を考慮した部品の順位付けを行う必要がある．

2.4.1 Keyword Rank 法

索引語の中には部品の内容と密接に関係したものもあれば，関係の薄いものも存在する．抽出された索引語が部品の内容を表すうえでどれだけの重要度を持っているか測ることができれば，より望ましく順位付けされた検索結果が提供できる．このために用いられるのが索引語の重み付けである．索引語の重みを利用することによって，同じ索引語を含む部品でもその索引語の各部品中での重要度を考慮して，検索キーに対する部品の適合度を計算し部品を順序付けすることが可能になる．検索者が与えた検索キーがある部品の索引キーにヒットしたとき，その索引キー中の索引語の重みの総和を検索キーと部品の適合度とする．SPARS-J における索引語の重み付け手法として，情報検索の分野で一般的に用いられる **TF-IDF** 法 [20] を用いる．TF-IDF 法は任意の部品中における特定の索引語の出現頻度 $TF(Term\ Frequency)$，および特定の索引語を含む部品数の逆数 $IDF(Inverse\ Document\ Frequency)$ の値を正規化して重みを算出する．TF は部品内で出現頻度の低い索引語と高い

索引語を差別化し，部品をより特徴付ける語を選別するためのものである．IDF は部品集合内の他の部品の索引語の分布について考慮するためのもので，ある索引語が，どの程度その部品に特徴的に現れるのかという特定性を示す．検索キーと部品の適合度の高い順に順位付けすることを，後述する CR 法に対して *KR* **法**(*Keyword Rank* **法**) と呼ぶことにする．また，測定した適合度の値を KR 値と呼ぶ．

SPARS-J では，索引語の重みを算出する際に，そのトークンの種類によって異なる重みを与えている．表 1 に区別する索引語の種類と重み付けの例を示す．どの種類の索引語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が，経験的にクラス定義名やメソッド定義名など，その部品の概念を象徴した名前が付けられる傾向の索引語に対して大きな重みを設定している．

索引語の種類	重み	索引語の種類	重み
クラス定義名	200	メソッド定義名	200
インタフェース名	50	パッケージ名	50
インポートパッケージ名	30	呼出しメソッド名	10
参照フィールド変数名	10	生成したクラス名	10
変数の型名	10	参照する変数名	1
コメント (/*...*/)	30	文書コメント (/**...*/)	50
行末コメント (//...)	10	文字列リテラル	1

表 1: 索引語の種類とその重み

部品を c，与える m 個のキーワードを $t_1, \ldots, t_m$，c 中の全ての索引語の種類の集合を $d(c)$，c 中の種類 s の索引語 t の出現回数を $tf_s(t, c)$，データベース中の総部品数を N，索引語 t を含む部品の数を $df(t)$ と表す．さらに，$kw(s)$ は索引語の種類 s の重みを指し，その値は表 1 に従う．例えば，kw(クラス定義名) は 200 である．そのとき部品 c の KR 値は以下の式で求める．

$$KR = \sum_{i=1}^{m} (log_e(\sum_{s \in d(c)} kw(s) \cdot tf_s(t_i, c)) \cdot \frac{N}{df(t_i)})$$

2.4.2 Component Rank 法

我々はこれまでに，利用関係からソフトウェア部品の利用実績を測定し，順位付けする手法 (*Component Rank* **法，***CR* **法**) を提案している [13],[33]．CR 法では，十分な時間が経過し利用関係が収束した部品の集合に対して，各部品間に存在する利用関係に基づいてグラフおよび行列を構築し，構築された行列に対して繰り返し計算を行う事で各部品を評価する．求められる値は，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順位は高くなる．CR 法によって測定した適合度の値を CR 値と呼ぶ．以下，我々が提案した CR 法について説明する．

重みの定義

CR 法では，図 2 部品グラフ $G = (V, E)$ 上の個々の辺およ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．頂点の重みと，辺の重みを次のように定義する．

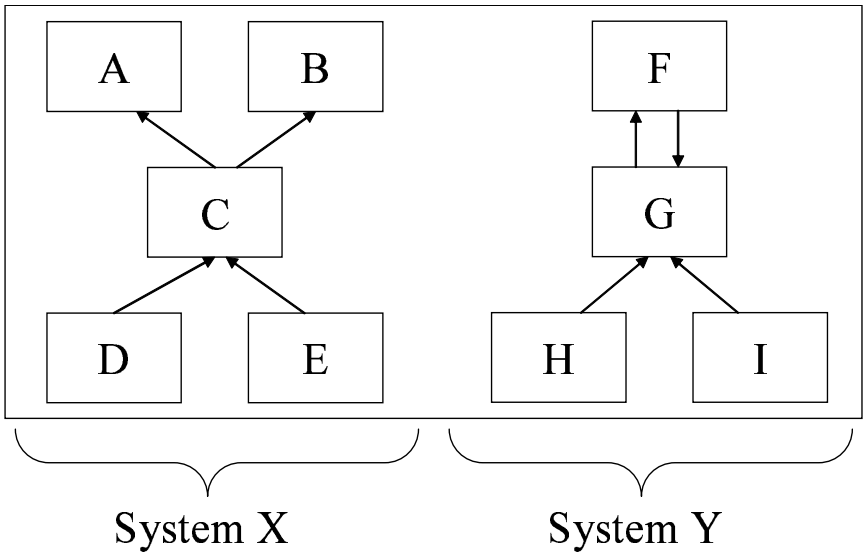

図 2: 部品グラフの例

[**定義 1**] 頂点の重みの和

部品グラフ G 上の各頂点 v は $0 \leq w(v) \leq 1$ の重みを持ち，G の頂点の重みの総和は 1 とする．すなわち，

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

[**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 $w'(e_{ij})$ は，

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

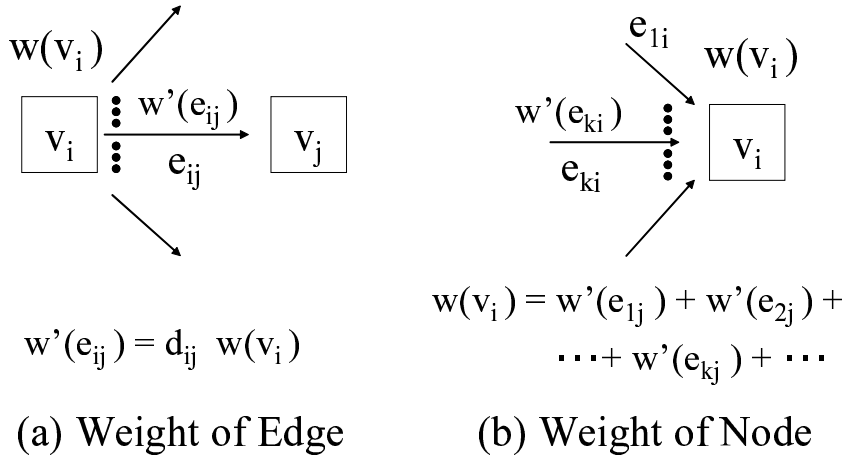

図 3: 重みの定義

図 3 (a) はこの定義を図示したものである．d_{ij} は配分率とよび，$0 \leq d_{ij} \leq 1$ 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 3 (b) は次の定義を図示したものである．

[**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．この時，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，

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

重みの計算

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．

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

類似度測定

似たような機能を提供する部品を類似部品という．例えば，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は類似している．ライブラリが様々なソフトウェアから構成される場合，その中にはコピーされたりコピーされた上で大小様々な変更が加えられた部品が少なからず存在し，ライブラリの規模が大きくなるほど類似部品は多くなる．

SPARS-J の類似度判定は [21] で提案された手法を用いている．[21] では，Java ソースコードから静的解析によって各クラスの静的特性メトリクスを抽出する．そして，それらのメトリクス値からクラスの類似度を判定する．

類似部品化の効果

それぞれの類似部品へ指されていた辺が，部品群化を行う事で一つの類似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，その部品群は多くの部品から利用されることになる．このため，コピーされたという利用実績を重要度に反映させる事ができる．

この時，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 4**] 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 4 は部品群化を行った際に，類似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．3 つのシステムのうち，2 つのシステムには $X.A$ および $Y.A$ という同一の部品が存在する．部品がコピーされた場合は，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．部品群化を行わないと，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 4 の場合すべての部品の重要度が等しくなる．そこで，類似部品群化によって部品 $X.A$ および $Y.A$ 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，部品 A の重要度が他の部品よりも高くなる．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．

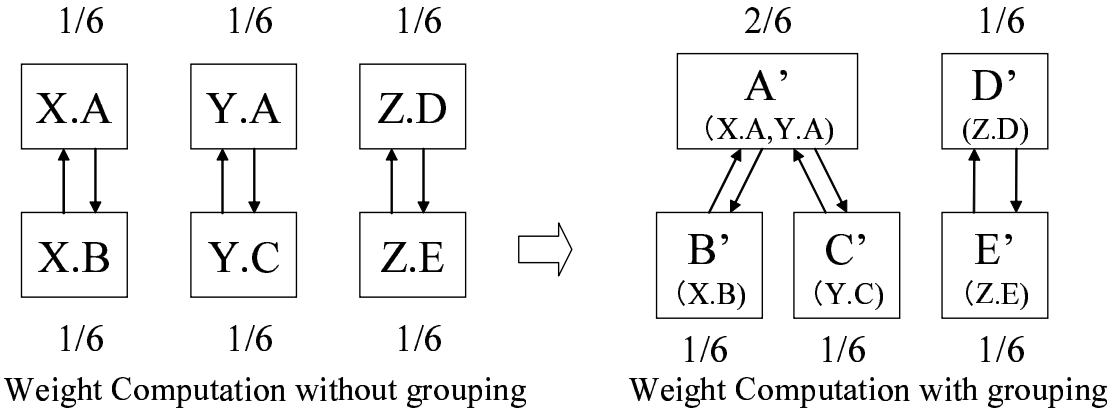

図 4: 部品群化の効果

2.4.3 順位の統合

CR 法によって得られた順位と KR 法によって得られた順位を統合する．順位の統合はメタ検索システムで利用される手法であり，経験的な (heuristic) なものとして知られている．メタ検索システムとは，複数の検索システムで同一の検索対象集合の検索を行ない，検索結果を統合するシステムである．検索システムそれぞれが持つ検索アルゴリズムの特徴を考慮した検索が可能であり，検索結果の再現率と適合率を改善することができる．順位の統合は，広告目的で作成者が検索結果で意図的に高い順位を得ようと作成した Web ページなど，不相応に高い順位を得ているもの (spam と呼ばれる) の順位を下げ，情報の健全性を守ることができるとされている [9]．最も単純な順位の統合手法に Borda の手法 [6] がある．計算量が少なく，線形時間で解けることが知られている．Borda の手法では，各順位に対して評価点を割り当て，その合計値をもとに最終的に統合された順位を求める．

SPARS-J では，検索結果を CR 法，KR 法で評価し，それぞれの順位を部品の評価点とする．そして評価点を足した合計点によって検索結果を昇順に並べ直し，検索結果として提供する．例を用いて Borda の手法を説明する．表 2 は CR 法，KR 法，および Borda の手法で統合された結果 (CR+KR) を，表 3 はそれに対応した各部品の評価点を表す．表中の A, ···，J はそれぞれ部品である．例えば部品 A は CR で 1 位，KR で 4 位であり，評価点は 5 点となる．よって部品 A は順位を統合した結果 2 位となる．

	CR	KR	CR+KR
1 位	A	C	C
2 位	E	F	A
3 位	C	G	E
4 位	I	A	F (3 位)
5 位	J	B	B
6 位	B	D	I (5 位)
7 位	H	I	G
8 位	F	E	J
9 位	G	H	D
10 位	D	J	H (9 位)

表 2: Borda の手法

	CR	KR	CR+KR
A	1	4	5
B	6	5	11
C	3	1	4
D	10	6	16
E	2	8	10
F	8	2	10
G	9	3	12
H	7	9	16
I	4	7	11
J	5	10	15

表 3: 評価点

2.5 検索結果

以下に SPARS-J のスクリーンショットを載せる．図 5 は SPARS-J のトップページを表しており，ここから検索要求を検索キーワードという形で与えることでソフトウェアを検索することができ，図 6 のように順位付けされた検索結果一覧が得られる．個々の部品を選択すると図 7 のように検索キーワードがハイライトされた部品が表示され，利用関係や類似部品，メトリクス値等の詳細情報を取得することができる．図 8 はパッケージブラウザであり，図 7 に加えて，クラスの階層構造や部品内のメソッド情報も併せて表示される．

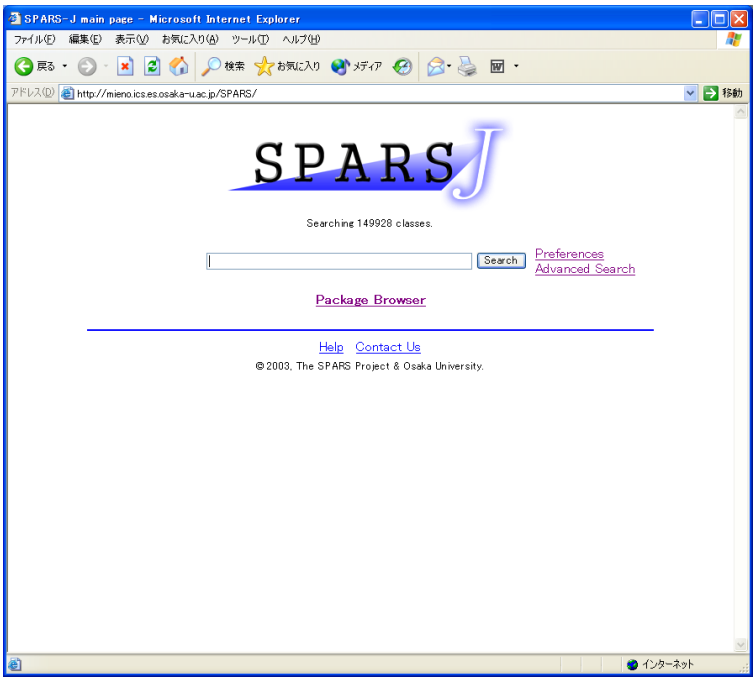

図 5: 検索画面

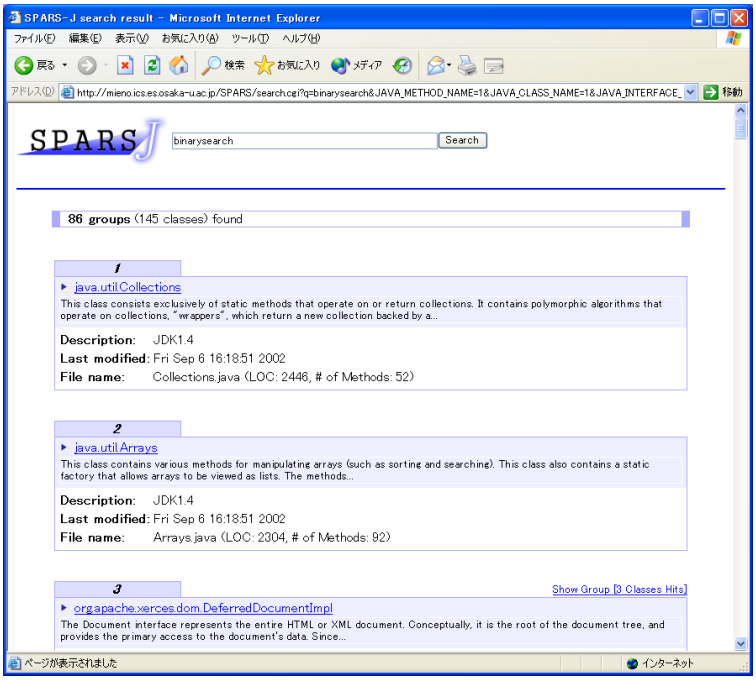

図 6: 検索結果一覧

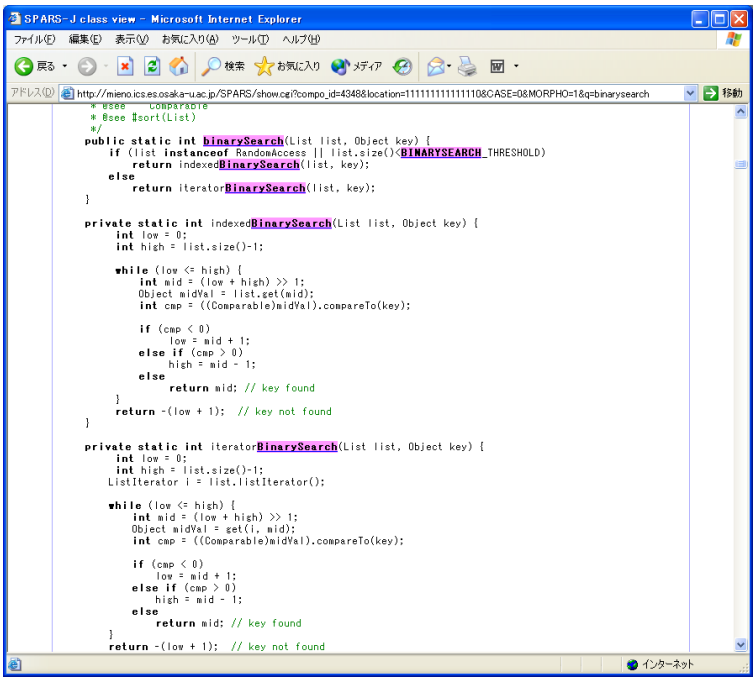

図 7: ソースコードのハイライト表示

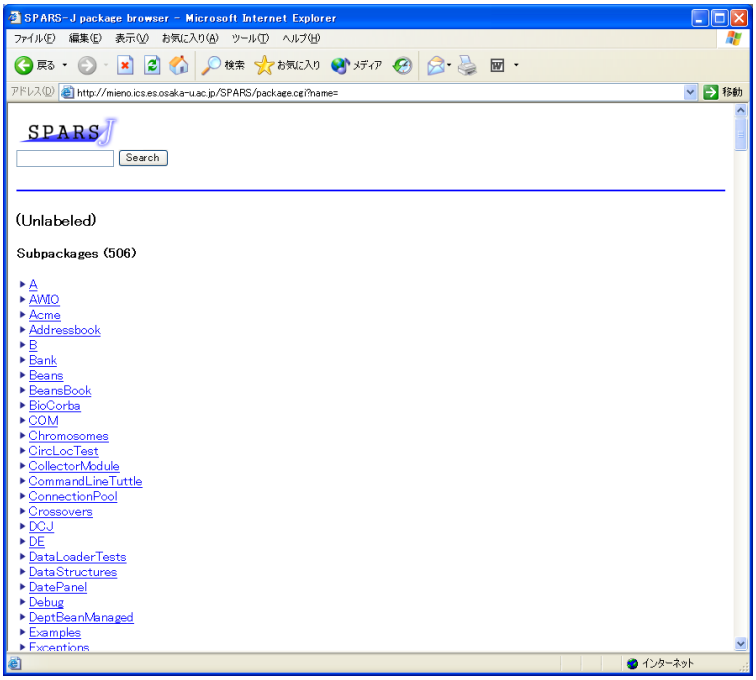

図 8: パッケージブラウザ表示

2.6 関連研究

これまでにソフトウェア部品検索を行うために様々な手法が提案されている．

WWW のソフトウェア部品を自動収集し，それらを解析することでデータベースに分類・蓄積し，作成したデータベースを対象に検索を行う形式の検索システムに [1],[17],[18] がある．Agora[1] はサーチエンジン AltaVista の SDK を利用してソフトウェア部品の自動収集を行う．CORBA オブジェクトである ORB や JavaBeans を検索可能である．jCentral[17] は IBM が開発した Java のソフトウェア部品検索システムである．jCentral では Java ソースコードの他，アプレット，JavaBeans，ニュース，FAQ，チュートリアルなどの情報を検索することが可能である．亀井ら [18] の提案したシステムでは，クエリにソフトウェアメトリクスなどの特有の値を含めてソフトウェアを検索することができる．

庭山ら [24] はセマンティック Web の手法を利用してソフトウェアのソースコードにメタデータを付与し，検索ワードをオントロジ変換したものとのマッチングを行うことで，ソフトウェア部品を意味に基づいて検索する手法を提案している．また，鷲崎ら [34] は独立して再利用可能なコンポーネントを，検索および試行可能な検索システムの提案を行なっている．提案システムでは，Java ソースコードからクラス間の依存関係を解析することで，要求された機能を実現する複数のソースコードを JavaBeans コンポーネントとして抽出し，ユーザに提供する．

その他，ソフトウェア部品検索として用いることが可能な自然言語文書検索システムとして，日本語全文検索システム Namazu[22] や Web ページ検索システム Google[11] などが存在する．

3　評価実験

3.1　実験にあたって

3.1.1　実験目的

2 節で我々の研究グループが開発している SPARS-J について説明した．SPARS-J では，ソースコード内のクラスをソフトウェア部品とみなして，利用関係や類似といったソースコード特有の特性を考慮しながら大量のソフトウェアを自動的に関連付けし，検索キーに関連したソースコードの効率的な検索機能を実現している．さらに，検索結果表示の際にソフトウェア部品に関する詳細な情報を併せて提供する．このシステムを用いることで，ライブラリの知識が無い開発者も有用なソフトウェア部品やそれに付随する有益な情報を平易に入手できることが期待される．

しかし，SPARS-J のソフトウェア部品検索システムとしての有効性についての評価は行われておらず，システムの有効性の評価が必要である．

そこで，本節では SPARS-J について，ソフトウェア部品検索システムとしての実験的評価を行い，システムの有効性を検証する．

実験では以下のことについての評価を行う．

1. 順位付けの性能について一般的な検索システムとの比較
   ソフトウェア開発者が過去に同じように開発されている部品が存在すると認識している場合，プログラミング時に既存のソースコードや過去の有用なソースウェア部品を参照しようとするものである．この評価においては，一般的な検索システムと比較して，SPARS-J が該当する部品をその部品の重要度と併せて，どれだけ十分開発者に提供できるかを評価する．SPARS-J を用いることで迅速に目当てのソフトウェア部品を検索でき，ソフトウェア開発に SPARS-J が有効であることを示す．

2. SPARS-J 内で利用される各順位付け手法の比較
   2.4 節で示したとおり，SPARS-J においては，CR 法と KR 法の 2 通りの順位付け手法があり，更にそれらを統合した CR+KR という順位付けを行うことができる．この評価では各順位付けの特徴を調査し，それらの中でどの順位付けが優れているのか，各順位付けがどのような点で優れているのかを評価する．

3. 企業内のソフトウェアに対する SPARS-J の適用
   SPARS-J はただ単に検索機能を提供するだけでなく，データベース内における利用関係や類似部品等，部品に関する様々な情報を提供している．この評価では，それらの情報が実際の開発現場ではどのように役立てられるのかを評価する．

実際の実験では，1 に関しては，比較対象の検索システムとして，Web ページ検索システムである Google[11] と全文文書検索システムである Namazu[22] を選択した．Google は非常に大規模な検索システムであるため，ジャンル問わず多くの検索目的で利用可能であり，また Namazu はフリーウェアでありながら多方面で利用されている信頼性の高い検索システムである．

1，2 に関しては，JDK1.4 に加えて，インターネット上で公開されているオープンソースなプロジェクトにおけるソースコードを収集し，収集した約 14 万クラスに対してデータベースを構築した上で，評価実験を行った．

また，3 に関しては，企業内のソフトウェア部品約 2500 クラスを対象としてデータベースを構築して評価実験を行った．

以下では評価における指標について説明をした後に各実験内容についてその詳細と実験結果を示す．

3.1.2 実験概要

一般に，検索システムの性能を評価するときによく用いられている手法には，**適合率**(*precision ratio*) と**再現率**(*recall ratio*) を求めて評価する方法がある．

適合率

検索された文書の中で，適合する文書数の割合

$$\text{適合率 } P = \frac{\text{検索された文書の中で適合する文書数}}{\text{検索された文書数}}$$

再現率

データベース中のすべての適合する文書の中で検索された文書数の割合

$$\text{再現率 } R = \frac{\text{検索された文書の中で適合する文書数}}{\text{データベース中の適合する全文書数}}$$

再現率を求めるためには，データベース中の全適合文書数が必要であるため，実験的環境でしか算出できない．また，大規模なデータベースで，各検索要求に適合する文書を網羅的に調べるには多くの労力が必要であり，非常に困難である．また，現在，文書検索システムでは，利用者が入力した問合せに適合している順に検索結果が表示される「適合度順出力」が主流である．このような検索システムでは，検索実験を通じて最適なパラメータを設定する必要があり，文書検索システムの研究・開発には，検索実験に使用できるテストコレクション [25] が不可欠である．

文書検索システムを対象とした性能評価では，BMIR-J2[5] 等のテストコレクションが利用可能であるものの，ソフトウェア部品検索システムを対象としたテストコレクションは現

在のところ存在していない．そこで本実験では，再現率の代わりに別の尺度で評価することにする．

ソフトウェア検索システム SPARS-J の順位付けの性能評価を行うために，本実験では様々な検索キーワードに対する検索結果に対して，

- その中にどれだけ有用なソフトウェア部品が含まれているか
- システムによる順位付けがユーザによる順位付けにどれだけ近似しているか

という 2 種類の観点から評価を行う．

一般に，情報検索システムが検索した結果において，その文書数がいくつであれ，その文書全てを閲覧するようなことはしない．例えば検出文書が 100 件あったとしても実際に閲覧される文書というのはその中の上位 10 件程度であり，その中に目的文書を見つけられなかった場合は，更に以降の文書を閲覧するよりは，検索キーワードを変えて再度検索し直す傾向がある [2]．また，目的文書が見つかった場合にも，その時点で閲覧することをやめるということが通例である．そのため，ソフトウェア検索システムにおいては，全検索部品に対しての適合部品の割合より上位 10 件以内の部品にどれだけ重要な情報が詰まっているかの方がはるかに意味があることだと考えられるので，本実験では検索結果の上位 10 件のソフトウェア部品を対象として評価を行うことにした．

評価尺度

- 適合率
- ndpm(*Normalized Distance-based Performance Measure*) 法 [32] による評価

検索結果の上位 10 件中にどれだけ有用なソフトウェア部品が含まれているかを比較するために，上位 10 件の適合率を比較し，システムによる順位付けがユーザによる順位付けにどれだけ近似しているかについては，上位 10 件の部品に対してユーザ・プリファレンスの概念を用いた ndpm 法 [32] を適用してその評価値を得る．以降，ndpm 法によって測定された評価値を ndpm 値または単に ndpm と呼ぶ．

ndpm 法 [32]

ndpm 法とはユーザ・プリファレンスの概念を用いて，文書間の関係や有用性等の計測を行ったものである．すなわち，ある文書とある文書を比較して，ユーザがどちらの方が適合している文書であるかを判断し，評価したものである．文書に対するユーザの判断というものは，相対的な順序付けにより形式的に説明でき，順序的尺度で計測できる．このフレーム

ワークにおいて，ユーザの順位付けとシステムの順位付け間の距離に基づくシステムの性能を計測する新しい手法が提案されている．この手法では，文書間の双対的な順序のみを使用し，それによって，有用な順序的尺度の測定を確証している．さらに，それは様々なレベルの関係を判断すること，そして，評価システムの出力に使用可能である．

適合率の評価では，検索システムの評価を**適合している/適合していない**の 2 つの値でしか評価できていない．複数値の妥当性尺度を用いることの難しさは，そのような尺度をどのようにして設計するかが明確でなく，また，有限で固定された評価尺度の採用は，ユーザの判断を計測したり，反映したりするには，不適当であるかもしれないという指摘もされている．そのため，ある制限のない尺度が使われるべきであり，ユーザはあらかじめ決められた複数の値の集合を使用することを強いられるべきではないと，提案されている．そこで [32] におけるフレームワークの中で，文書が妥当か，そうでないか，を述べることに代わり，ユーザはある文書が他の文書より有用か，そうでないか，を明示する．これは，ユーザはある文書と他のものの文書のどちらを好むか，ということである．この計測法では，順位付け中の文書の相対的な順序のみを使用する．

ndpm 値の計算方法

文書のユーザプリファレンスは，文書を二つ一組にして比較することで説明することができる．文書集合の中から二つを取り出したとき，ユーザはある文書がもう一方の文書より有用である，あるいは適合する文書であると決めることができる，と仮定する．D を文書の有限集合を表すとする．ユーザプリファレンスは，$d, d' \in D$ について，D 上の二項関係 $\succ$ によって定義される．

ユーザプリファレンスの定義

ある文書のペア d, d' において，

$$d \succ d' \Leftrightarrow \text{ユーザは } d' \text{より } d \text{の方を好む}$$

$\succ$ 関係は (厳密な) プリファレンス関係と呼ばれ，デカルト積 $D \times D$ の部分集合である．

プリファレンス関係

ある文書のペア d, d' において，

$$\succ = \{(d, d') | \text{ ユーザは } d' \text{より } d \text{の方を好む } \}$$

$\Gamma(D)$ を文書集合 D における全ての順序付けの集合とすると，順位付け間の距離を計測する方法として実数関数 $\beta : \Gamma(D) \times \Gamma(D) \rightarrow \mathbf{R}$ を用いる．

順序付け間の関係

ある一組の文書 $d, d' \in D$ における順位付け間の関係を以下のように呼ぶことにする．

- d と d' の順位が二つの順位付けの間で同じであれば，その順序付けは**一致である**という．
- 一方の順位付けでは d の方が順位が高く，もう一方の順位付けでは d' の方が順位が高くなっているのであれば，その順序付けは**矛盾である**という．
- 一方の順位付けでは d か d' のいずれかの順位が高く，もう一方の順位付けでは d と d' が同じ順位であれば，その順序付けは**互換である**という．

順序付け距離の定義

ある一組の文書 $d, d' \in D$ における順位付け間の距離を以下のように定義する．

$$
\begin{aligned}
&d \text{ と } d' \text{において} \succ_1 \text{ と } \succ_2 \text{ が一致であるとき，} && \delta_{\succ_1,\succ_2}(d, d') = 0 \\
&d \text{ と } d' \text{において} \succ_1 \text{ と } \succ_2 \text{ が互換であるとき，} && \delta_{\succ_1,\succ_2}(d, d') = 1 \\
&d \text{ と } d' \text{において} \succ_1 \text{ と } \succ_2 \text{ が矛盾であるとき，} && \delta_{\succ_1,\succ_2}(d, d') = 2
\end{aligned}
$$

[**例 1**] 文書集合 $\{d_1, d_2, d_3, d_4\}$ において，システムによる順位付け $\succ_s$ とユーザによる順位付け $\succ_u$ を考える．

$$
d_1 \succ_s d_2 \succ_s \begin{matrix} d_3 \\ d_4 \end{matrix}
$$

$$
d_2 \succ_u \begin{matrix} d_1 \\ d_3 \end{matrix} \succ_u d_4
$$

これらの 2 つの順位付けの間で，次が得られる．

$$
\begin{aligned}
&\delta_{\succ_s,\succ_u}(d_1, d_2) = 2 \\
&\delta_{\succ_s,\succ_u}(d_1, d_3) = 1 \\
&\delta_{\succ_s,\succ_u}(d_1, d_4) = 0 \\
&\delta_{\succ_s,\succ_u}(d_2, d_3) = 0 \\
&\delta_{\succ_s,\succ_u}(d_2, d_4) = 0 \\
&\delta_{\succ_s,\succ_u}(d_3, d_4) = 1
\end{aligned}
$$

距離関数

二つの順位付けの全体的な距離は，次のように計算できる．

$$\beta(\succ_1, \succ_2) = \sum_{d,d'} \delta_{\succ_1,\succ_2}(d, d')$$

これは全ての文書の組みについて計算されるものであり，明らかに，二つの順位付け間の距離は文書間の関係だけに依存している．

[**例 2**] 例 1 で用いた文書集合に対して順位付け間の距離を求める．

$$\delta_{\succ_s,\succ_u}(d_1, d_2) = 2 \quad , \quad \delta_{\succ_s,\succ_u}(d_1, d_3) = 1$$
$$\delta_{\succ_s,\succ_u}(d_1, d_4) = 0 \quad , \quad \delta_{\succ_s,\succ_u}(d_2, d_3) = 0$$
$$\delta_{\succ_s,\succ_u}(d_2, d_4) = 0 \quad , \quad \delta_{\succ_s,\succ_u}(d_3, d_4) = 1$$

これらの値から，$\succ_s$ と $\succ_u$ の間の順位付けの距離は次で与えられる．

$$\beta(\succ_s, \succ_u) = 2 + 1 + 0 + 0 + 0 + 1 = 4$$

ここで，同順位に順位付けされた文書を任意に再配列することで，容認可能な順位付けが実現される．この容認可能な順位付けの下では，システム性能は同順位に順位づけされた文書をどう順位付けするかということとは無関係に評価される．この原則によると，システム性能は，$\succ_s$ あるいは $\succ$ s に近い順位付けと，理想的な順位付けである $\succ_u$ はあるいは $\succ_u$ の容認可能な順位付けとの違いにより計測される．Rocchio[27] により，検索システムは容認できる順位付けを生成するためのものであると明白に述べており，実際に適合率・再現率といった多くのシステム性能の計測はこの原理に基づいている．このことは，$\succ$ s と $\succ$ u のある容認可能な順位付けとの間の距離を利用することで性能の評価が行えるであろうと示している．$\succ_u$ に関しては多くの容認可能な順位付けが存在するが，有効性の観点からこのような全ての容認可能な順位付けは等価である．公正な計測の定義として，$\succ_s$ にもっとも近い，ある容認可能な順位付けが選ばれるべきである．そこで，$\Gamma_u(D)$ は全ての容認可能な順位付けを意味するとすると，以下の **distance-based performance measure**(dpm).

例 3 (容認可能な順位付け) 例えば,

$$d_1 \succ_1 d_2 \succ_1 \begin{matrix} d_3 \\ d_4 \end{matrix}$$

という順位付けにおいて容認可能な順位付けとは

$$d_1 \succ_1 d_2 \succ_1 d_3 \succ_1 d_4 \text{と } d_1 \succ_1 d_2 \succ_1 d_4 \succ_1 d_3$$

がある．

提案されている dpm

$$dpm(\succ_u, \succ_s) = min_{\succ \in \Gamma_u(D)} \beta(\succ, \succ_s)$$

ここで $\succ_a$ を $\succ_u$ に最も近い容認可能な順位付けだとすると，dpm は以下で定義される．

dpm の定義

$$dpm(\succ_u, \succ_s) = \beta(\succ_a, \succ_s)$$

例 4 (dpm 値の計算)

$$\begin{matrix} d_1 \\ d_2 \end{matrix} \succ_u d_3 \succ_u \begin{matrix} d_4 \\ d_5 \end{matrix}$$

を，5つドキュメントの集合 $D = \{d_1, d_2, d_3, d_4, d_5\}$ における，ユーザの順位付けとし，

$$\begin{matrix} d_1 \\ d_5 \end{matrix} \succ_s \begin{matrix} d_2 \\ d_4 \end{matrix} \succ_s d_3$$

を，情報検索システムによって順位付けされた順位付けとする．このとき，$\succ_u$ に関して，$\succ_s$ に最も近い容認できる順位付けは以下で与えられる．

$$d_1 \succ_a d_2 \succ_a d_3 \succ_a d_5 \succ_a d_4$$

性能検索式に基づき，我々は以下を得る．

$$dpm(\succ_u, \succ_s) = \beta(\succ_a, \succ_s) = 8$$

ここで，この dpm とは絶対的な距離関数と考えられ，これを正規化された距離計測とするために **a normalized distance-base performance measure**$(ndpm)$ が以下のように定義されている．

ndpm の定義

$\Gamma(D)$ を容認可能な順位付けの集合とすると

$$ndpm(\succ_1, \succ_2) = \frac{dpm(\succ_1, \succ_2)}{max_{\succ \in \Gamma(D)} dpm(\succ_1, \succ)}$$

ここで，$max_{\succ \in \Gamma(D)} dpm(\succ_1, \succ)$ は D における全ての順位付けと $\succ_1$ の間の距離の最大値を意味する．直感的に検索システムによって提供される最も悪い順位付けとは，ユーザとは全く逆の順位付けである．すなわち

$$d_1 \succ_1 d_2 \succ_1 d_3 \text{に対して } d_3 \succ_2 d_2 \succ d_1$$

となる順位付けである．このとき全ての $d, d' \in D$ に対して $\succ_1$ と $\succ_2$ は矛盾しているから，n を D の要素数とすると，

$$dpm(\succ_1, \succ_2) = {}_nC_2 \times 2 = n(n-1)$$

となる．上の例では，

$$dpm(\succ_1, \succ_2) = 3 \times (3-1) = 6$$

となる．すなわち ndpm 値は以下のようにして計算できる．

ndpm の計算方法

n を文書集合 D の要素数とすると

$$ndpm(\succ_1, \succ_2) = \frac{dpm(\succ_1, \succ_2)}{n(n-1)}$$

[**例 5**] 例 4 で ndpm を求めると，$n = 5$ より，

$$\begin{aligned} ndpm &= \frac{dpm(\succ_u, \succ_s)}{5 \times (5 - 1)} \\ &= 0.4 \end{aligned}$$

となる．このようにして求められる ndpm 値を用いて評価を行う．

3.2 評価実験 1

3.2.1 実験準備

データベース

対象とした SPARS-J のデータベースは，Sun Microsystems[30] が提供している JDK1.4 に加えて，SourceForge.net[29] や The Apache Software Fundation[3] などのオープンソー

ス開発コミュニティから入手したソースコードから O'Reilly Media[26]，技術評論社 [10] といった出版社の WEB 上で公開されている出版物のサンプルプログラムのソースコードまで幅広く集めたソフトウェア部品群から構築されており，約 14 万ファイル (約 15 万クラス) が含まれている．このデータベースにおける各部品の CR 値の分布を図 9 に示す.

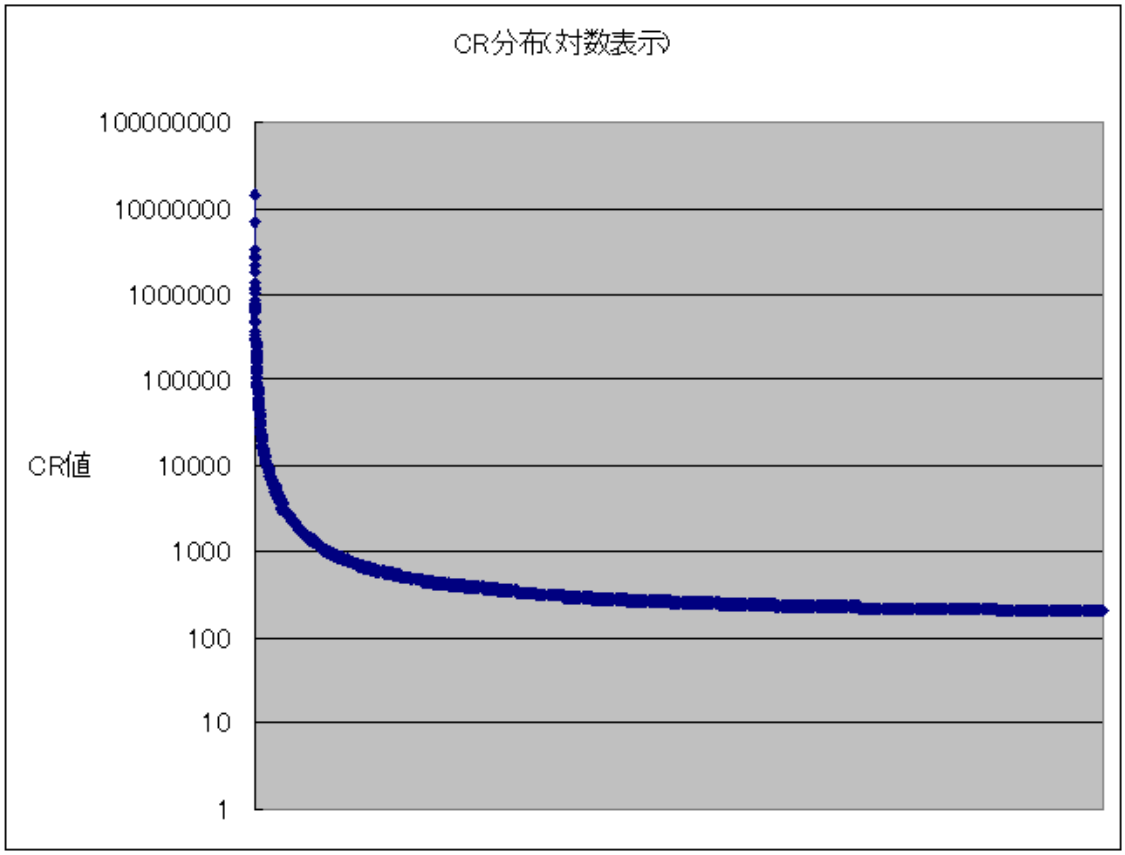

図 9: 構築したデータベースの CR 分布図

上記のデータベースの SPARS-J に適用する検索キーワードとその検索目的を表 4 に示す.

表 4: 検索キーワード

	検索目的	検索キーワード
K1	クイックソートメソッドのアルゴリズムを調べる	quicksort
K2	二分探索を行うメソッドのアルゴリズムを調べる	binarysearch
K3	アナログ時計のアプレットを作成するために関係する部品を探す	clock applet
K4	アプレットでテキストエリアを表示させる方法を調べる	applet textarea
K5	ランダムの数字を生成するメソッドのアルゴリズムを探す	randum number generate
K6	スタックを実装するために関係する部品を探す	stack push pop
K7	簡単なチャットシステムを実現するために関係する部品を探す	chat server client
K8	クラスファイルのダンプツールを作成するために関係する部品を探す	classfile dump
K9	zip ファイルを圧縮するプログラムを作成するために関係する部品を探す	zip deflate
K10	ストリームを介した入出力の適用例を調べる	write read inputstream outputstream

3.2.2 実験プロセス

表 4 に示した検索キーワードで検索した結果を評価する．具体的には，各システムの検索結果上位 10 件の部品に対して適合率を求め，比較を行う．

SPARS-J と他の検索システムとの比較

ここでは，SPARS-J による検索結果とその他検索システムの検索結果の比較を行う．比較の対象となる検索システムとして，以下の 2 つの検索エンジンを用いた．

- Google[11]
- Namazu[22]

Google はジャンルを問わず多くの検索目的に用いられており，Web ページ検索に限らずソフトウェア部品の検索にも用いることができ，実際にソフトウェアを検索するときには Google を用いることが多い．また，Namazu はフリーウェアでありながら官公庁，市役所，企業，

大学など幅広く導入実績から信頼性の高いシステムと言える．加えて，Namazu のデータベースに登録するソフトウェア部品を SPARS-J のデータベースに登録するソフトウェア部品と全く同じものにすることができ，純粋に検索システムの順位付け性能を比較することができる．

Namazu のスコア計算アルゴリズムは，SPARS-J の KR 法同様に TF-IDF 法に基づいて計算されている．しかし，Namazu では Java 言語で書かれているソースコードの構文解析は全く行われていないため，索引語の種類による重み付けは一切考慮されておらず，全ての索引語の重みが 1 として計算されている．そのため，SPARS-J の KR 法による順位付けは Namazu と比較して良い検索結果が得られると推測できる．

なお，SPARS-J における検索結果は類似部品を一つのグループにまとめているためそれらの部品群は一件の検索結果として扱われている．そのためある一つの部品が検索結果の上位に順位づけされている場合，その類似部品はいくつあったとしても順位付けに影響しない．そこでそれに対応した処置として，Google ではインデント表示されている類似ページは一件とし，Namazu の検索結果は手作業で類似部品をまとめ，それを一件として扱った．

また，Google を用いた検索では，表 tab:keyword で示しているキーワードに”java source” という語を付け加えて検索した．

3.2.3 実験結果

表 4 に示された各キーワードの検索結果に対する適合率の計算結果を表 5 に示す．表 5 中の K1,···,K10 は表 4 の検索キーワードの番号に対応しており，SPARS-J(CR)，SPARS-J(KR)，SPARS-J(CR+KR) はそれぞれ SPARS-J の CR，KR，CR+KR による順位付けの実験結果を表している．

表 5: 適合率の評価結果

	SPARS-J(CR)	SPARS-J(KR)	SPARS-J(CR+KR)	Google	Namazu
K1	1	1	1	0.7	0.9
K2	1	1	1	0.4	0.6
K3	0.5	0.5	0.5	0.3	0.4
K4	0.4	0.9	0.8	0.3	0.6
K5	0.4	0.4	0.4	0.1	0.3
K6	0.2	0.2	0.2	0	0.1
K7	0.9	1	1	0.3	0.4
K8	1	0.8	1	0.1	0.2
K9	0.6	0.7	0.7	0.4	0.4
K10	0.5	0.7	0.7	0.4	0.7
Ave.	0.65	0.72	0.73	0.3	0.46

3.2.4 分析・評価

適合率について SPARS-J の各順位付け (CR による順位付け，KR による順位付け，CR と KR を統合した順位付け) と Google,Namazu とで，対応のある平均値の差の検定 [12],[28] を有意水準 5%で行ったところ，SPARS-J の全ての順位付けが Google，Namazu との間に有意な差があると判定された．このことから SPARS-J は，ソフトウェア検索という観点から，総合検索エンジンや文書検索システムと比較して，より多くの部品をより的確にユーザに提供すると考えられる．

3.3 評価実験 2

3.3.1 実験準備

この実験では，実験 1 で用いたデータベースと検索キーワードをそのまま用いる．

3.3.2 実験プロセス

ここでは，SPARS-J におけるそれぞれの順位付け手法 CR，KR，CR+KR の比較を行う．CR 法はデータベース内のソフトウェア部品群内における各ソフトウェア部品の利用実績に基づく評価値により順位付けされ手法であり，KR 法はユーザが与えた検索キーワードに合致する索引キーを保持しているソフトウェア部品を適合度の高い順に順位付けする手法であるので，これら二種の順位付け手法は全く異なる観点から部品を順位付けする手法である．そこでこれらの順序付けの特徴を調査するために各順位付けについても評価を行う．また，

これらの順位付けを統合したCR+KRによる順位付けの評価も行う．CR+KRによる順位付けでは，CRとKRの両方の順位付けで高い順位を得られた部品が上位に順位づけされ，順位付けの性能が向上すると期待している．

具体的な内容は，実験1で用いた表4の各検索キーワードに対し，SPARS-Jの各順位付け手法で検索を行い，ユーザの想定する理想的な順位付けとそれぞれの検索結果とのndpm値を求め，ユーザの順位付けと各順位付け手法との違いを比較する．その結果を評価実験1で求められている適合率と併せて評価する．

3.3.3 実験結果

表4に示された各キーワードの検索結果に対するndpm値の計算結果を表6に示す．表中のK1,···,K10は表4の検索キーワードの番号に対応している．

表 6: ndpm法による評価結果

	SPARS-J(CR)	SPARS-J(KR)	SPARS-J(CR+KR)
K1	0.036078	0.048104	0.037004
K2	0.193676	0.260840	0.221344
K3	0.133333	0.116667	0.091667
K4	0.122908	0.199672	0.189121
K5	0.207827	0.191633	0.194332
K6	0.183755	0.184040	0.159717
K7	0.080519	0.103247	0.079870
K8	0.047059	0.109244	0.052101
K9	0.190476	0.304762	0.257143
K10	0.209524	0.323810	0.266667
Ave.	0.143373	0.178014	0.140611

3.3.4 分析・評価

適合率において，対応のある平均値の差の検定を有意水準5%で行ったところ，KR，CR+KRがCRと比較して優れているという有意差が見られた．また，ndpm値についても適合率同様，対応のある平均値の差の検定を行ったところ，CRとCR+KRが，KRと比較して有意水準5%で有意な差が見られた．これらのことから，CRはKRと比較して，よりユーザが好む部品を上位に順位付けしていると考えられ，KRは全体的に適合している部品を上位に順位付けしていると考えられる．また，CR+KRはCRとKRの両方の性質を受け継ぎ，それぞれCR,KRより順位付けの性能が改善していると考えられる．これは，CRかKRの片

方の順位付けでのみ高順位に順位付けされても CR+KR による順位付けでは上位には順位付けされず，CR と KR の両方で高順位に順位付けされた部品のみが上位に順位付けされるからであると考えられる．

3.4 評価実験 3

3.4.1 実験準備

ここでは，某食品製造企業の協力を得て，SPARS-J を実際の開発現場のソフトウェアに適用した．

被験者

某食品製造企業内のソフトウェア開発者および管理者，合わせて 10 名．企業内のデータベースに含まれているソフトウェア部品についての知識がある程度あり，検索結果の部品に対しての適合/不適合の判断ができる．

データベース

企業内のソフトウェア部品の一部 (約 2400 クラス) を用いて SPARS-J のデータベースを構築し，評価実験を行う．図 10 はこのデータベースにおける各部品の CR 値の分布である．

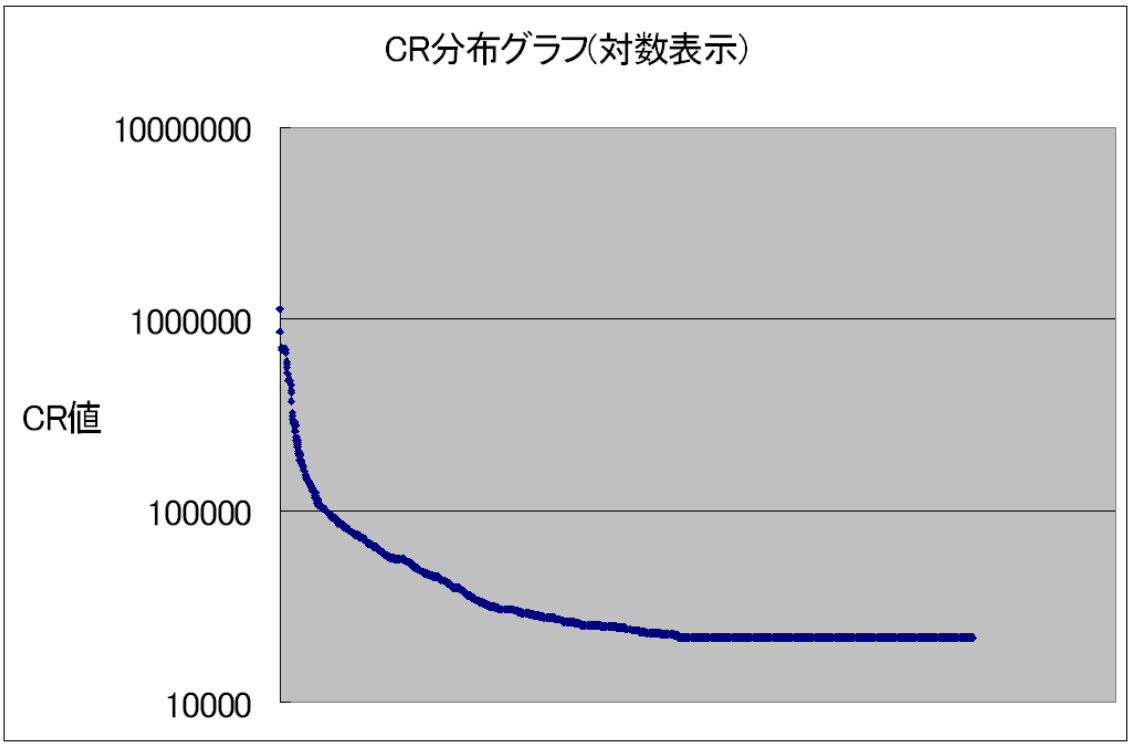

図 10: 企業内ソフトウェアの CR 分布図

3.4.2 実験プロセス

10 名の被験者に対して，実際の開発・保守時に SPARS-J を用いてもらいそれぞれ一つずつ検索を行う．

表 7: 企業内の実験に用いた検索キーワード

検索者	検索目的	検索キーワード
A	販売先取得メソッド (販売先コンポ) の変更による影響範囲の把握	getHambaisaki
B	isAdministrator というメソッドが必要なさそうだったので，削除しても影響がないかどうか調べたい	isAdministrator
C	某システムにおいて，エラーメッセージをプロファイル化するにあたり，目標のエラーメッセージに対応する Exception が使用されている箇所を漏れなく探す	ShoriStatus-AlreadyChanged-Exception
D	どのような check 系メソッドが実装されているか調べたい	check
E	スケジュールチェックをするメソッドを探す	schedule
F	テーブル (BTA***) を使用しているコンポを調べたい	BTA 002
G	カナ名を get するというメソッドのあるコンポを調べたい	get Kana
H	SGNCompanyCode の取得の仕方を調べる	SGNCompanyCode
I	ユーザの漢字氏名の取得の仕方を調べる	KanjiShimei
J	Password チェックの仕方を調べる	Password

3.4.3 実験結果

表 7 の各検索キーワードの検索結果に対する適合率の計測結果を表 8 に，ndpm 値の計測結果を表 9 に示す.

表 8: 適合率の評価結果

	SPARS-J(CR)	SPARS-J(KR)	SPARS-J(CR+KR)
A	0.3	0.6	0.3
B	1	1	1
C	1	1	1
D	0.7	0	0.6
E	0.8	0.5	0.8
F	0.5	0.8	0.8
G	0	0.7	0.6
H	0.75	0.75	0.75
I	0.2	0.2	0.2
J	0.2	0.7	0.3
Ave.	0.506024	0.60241	0.614458

表 9: ndpm 法による評価結果

	SPARS-J(CR)	SPARS-J(KR)	SPARS-J(CR+KR)
A	0.266667	0.688889	0.377778
B	1	0.666667	0.666667
C	0.809524	0.523810	0.714286
D	0.066667	0	0.111111
E	0.555556	0.311111	0.222222
F	0.555556	0.222222	0.244444
G	0	0.066667	0.155556
H	0.214286	0.392857	0.392857
I	0.3	0	0.2
J	0.222222	0.466667	0.2
Ave.	0.399048	0.333889	0.328492

3.4.4 分析・評価

表 8 より，適合率の平均は CR+KR が一番優れており，次いで KR，CR の順となっている．しかし適合率について対応のある平均値の差の検定 [12],[28] を有意水準 5%で行ったところ，３種類の順位付けには有意な差は見られなかった．また，ndpm 値においても CR+KR が一番優れており，次いで KR，CR となっているが，適合率同様，対応のある平均値の差

の検定において有意水準 5%で有意な差は見られなかった．この原因の一つとして，索引語の切り分けの問題が挙げられる．本実験を行ったとき，企業内のソフトウェアに適用した SPARS-J は 2004 年 2 月現在のバージョンより古いバージョンであったため，ソースコードからの索引語の切り分け性能が劣っていた．それが原因で検索時に指定の検索キーワードでは検索者の意図したソースコードが検索されないという状況がいくつかあり，良い実験結果は得られなかった．実際，その時に索引語の切り分けの性能がもう少し上がれば欲しい部品が手に入るという意見をいただいた．しかし，現在はその問題に対応しており，実験結果も改善されると推測している．

3.5 アンケートによる評価

本システムの使いやすさに関して行ったアンケートの項目とその回答結果を下の表 10 に示す．回答は 1 から 5 までの 5 段階評価で，5 が一番良い評価であり，逆に 1 が一番悪い評価となっている．また表中の空白はその項目に対して回答されていなかったことを表している．また，表の一番右の列にはその項目において一番回答が多かった評価値 (最頻値) を表している．

表 10: アンケート内容とその回答

	回答 (良 5 4 3 2 1 悪)							
回答者	A	B	C	D	E	F	G	最頻値
使い勝手について								
パッケージブラウザ	5	4	4	5	5	3	3	5
類似部品のグループ化	5	2	4	5		4	3	5,4
被利用クラスの参照	5	5	5	5		5	5	5
利用クラスの参照	5	5	5	1		5	5	5
メトリクス値	4	2		1	1	4	5	4,1
ソースコードのダウンロード	3	5		1	5	2	5	5
時間的コストの削減	5	3		3	5	4	1	5,3
ソフトウェア品質の向上	3	3		5	3	4	1	3
企業内のソフトウェア把握	5	5	3	1	3	2	1	5,3,1
総合評価	5	4		4	5	4	2	4
今後も利用したいか	5	4		5	5	3	4	5
表示について								
検索結果一覧	5	3	4	4	5	3	5	5
ハイライト表示	5	5	3	5	5	5	5	5

表 11 は SPARS を利用していただいた従業員に自分の業務において，何らかのツールを用いて実現したいと考えていることと，SPARS-J を用いることで実現できると考えていることについてのアンケートの内容とその結果を示している．表中の数字はアンケートに回答していただいた 7 名のうち何人が実現したい/できると考えているかを意味している．

表 11: SPARS を使って実現したい/できると考えていること

アンケート内容	したい	できる
コンポーネントを利用しているアプリの把握	5	4
コンポーネント，アプリのメトリクスによる解析	2	2
項目の属性に変更があった場合の影響範囲の調査	6	2
コンポーネント改訂時の影響範囲の調査	6	5
ソースの品質チェック	4	4

表 10 のアンケート結果より，特に良い評価を得ていると考えられるものは，実際の開発現場の開発者や管理者は,「パッケージブラウザ (図 8)」,「類似部品のグループ化」,「利用クラスの参照」,「被利用クラスの参照」「検索結果一覧」,「ハイライト表示」である．中でもパッケージブラウザと利用クラスの参照，被利用クラスの参照という項目は，企業内ソフトウェアの保守という観点に密接に関係しており，非常に便利であると考えられる．またその他の意見として，検索時間が短いという意見もいただいている．次に，表 11 のアンケート結果から，SPARS-J を用いて実現したいと考えていて，かつ実現できることとして高い評価が得られているものは,「コンポーネントを利用しているアプリの把握」,「コンポーネント改訂時の影響範囲の調査」である．これらの項目も，企業内ソフトウェアの保守という観点と密接に関係している．以上のことから，SPARS-J は企業内での利用において，保守作業に対しても非常に有効であると言える．以下に企業の方からいただいたコメントを掲載する．

SPARS-J への期待

1. コンポーネント利用状況の可視化
   - 依存関係が把握できる
   - コンポーネント改修時の影響を詳細に確認できる
2. 全 Java 資産の横断的検索 “ソースサーフィン”
   - 一筆書きで実行ステップをたどれる
     途中に他人のソースが挟まっても追える
3. オープンソース的な品質の向上

- 他人のソースのバグに気付く機会が増える

4. リファレンスと実物の 100%一致
   - リファレンスを，常に実ソースから取得
5. ソースリバースのため，運用負荷がかからない
   - 管理情報の入力が不要

品質施策等への応用可能性

1. コンポーネント利用率調査
   コンポーネント指向開発を強めていくために，以下のような思想が導入できないか?
   - コンポーネントとアプリの色分け
   - 部品を軸とした利用状況マトリクス (あるいはツリー)
   - ソース単位 → 主観的な “機能” 単位
2. ソース品質調査
   - メトリクス情報 (ステップ数，コメント数等) を元に，クラス毎の複雑度や情報充実度を品質指標に活用
   - コードクローン解析ツール (CCFinder) の組み合わせで，ソースコピー等によるバグ波及の可能性 (バグが広がる危険性) を数値化できか?
3. 開発規模評価
   - SPARS-J が出力するメトリクス情報や，CCFinder のソースコピー情報を利用することで，開発規模のネット値 (補正後の評価値) を算出できないか?

実利用に向けたその他検討領域

1. 管理単位との関係
   - システムＩＤ，ユースケース，コンポーネント等のマネジメントで認識している単位との関係
2. 他管理対象との関連
   - インフラ (サーバ) 配置，データベース，サービス開閉 · · ·

3. セキュリティ(機密保持)

   - 全資産を全員に見せることはできない
   - ブラウズ項目とユーザグループの組み合わせ管理

4. リリース管理の視点 (バージョン)

   - バージョンの考え方をどう持ち込むか?

5. 開発途上資産と運用資産

   - 対象者の違い，検索視点の違い
   - 検索サイトを２つ用意すべき?

このようなことから，今回協力していただいた企業の方々は，まず企業内のソフトウェアの管理を第一の目標として SPARS-J を利用していると考えられる．

4 考察

4.1 SPARS-J と他の検索システムとの比較

実験結果より，SPARS-J は Java ソフトウェア部品検索時には Google や Namazu より優れていると言える．

Google は Web ページ検索システムであり，ソフトウェア部品検索に特化していない．そのため，検索対象範囲が広すぎて検索結果が絞りきれず，ソフトウェア部品検索として必要のないものが多数上位に順位付けされ適合率が高くないのだと考えられる．

Namazu は SPARS-J の KR 法と同様に評価値に TF-IDF 法を用いている．しかしながら，適合率において有意差が見られたのは先述した通り，SPARS-J では構文解析を行い，索引語の種類によって重み付けを行っているためであると考えられる．索引語の重み付けを行うことで重要な位置に出現している場合には高い評価を，重要でない位置での出現には低い評価しか与えないことで，語句の出現回数が多いだけの部品は上位に順位付けされなくなっている．

4.2 SPARS-J 内における各順位付け手法の比較

評価実験 2 においては，CR 法は ndpm 値の点で優れており，KR 法は適合率の点で優れているという結果を対応のある平均値の差の検定により有意水準 5%で有意差が見られたものの，評価実験 3 においては，有意差は見られなかった．この原因として，評価実験 2 では主にツールを開発するという目的で SPARS-J を用いたのに対し，評価実験 3 では主に企業内のソフトウェアを把握するということを目的として SPARS-J を用いたからであると考えられる．また，実験時の SPARS-J のバージョンでは，索引語の切り分け方の性能が高くなく，企業の方からは「索引語の切り分け方の性能が高ければ，もっと良い検索結果になっている」という意見をいただいた．評価実験 2 はこの問題に対応した SPARS-J で評価を行ったので結果に差異が生じているのだということも考えられる．

CR+KR については，評価実験 2 において CR 法，KR 法をそれぞれ単体で用いて順位付けした結果よりも対応のある平均値の差の検定により有意水準 5%で有意差が見られており，評価実験 3 においても有意差は見られなかったものの，平均値は単体の順位付けよりも向上している．これらの結果に，CR+KR は CR 法と KR 法の両方の性質を併せ持ち，両方の順位付けで高い順位を得られた部品が上位に順位付けされるということを考慮すると，CR 法，KR 法単体で順位付けを行うよりも統合して順位付けを行う方が順位付けの性能が向上するのだと考えられる．

4.3 SPARS-J の有効性

本実験より，SPARS-J はソフトウェア部品検索システムとして，高度な順位付けの性能を持ち，ソフトウェア開発時の部品再利用の促進・支援を行い，その結果，ソフトウェア品質の向上につながると考えられる．またソフトウェアの保守時にはパッケージブラウザを用いたり利用クラス/被利用クラスを参照することによりデータベース内の利用関係を可視化することができ，ソフトウェア理解につながると考えられる．

また，本システムはサーバ側にデータベースを構築してシステムを起動させるので，システム自体をクライアント側にインストール必要がなく，Web ブラウザさえ使用可能であれば本システムが利用でき，同時に多くのソフトウェアの資産を共有することができることになる．このような点からも SPARS-J がソフトウェア部品検索に便利なシステムであると言える．

このように，SPARS-J は開発にも保守にも利用することが可能で，かつ抜群のユーザインターフェースを持つことから，非常に優れたシステムであると言える．

5 むすび

本研究では，ソフトウェア部品検索システム SPARS-J の実験的評価を行った．SPARS-J と他の検索システム Google，Namazu との比較と SPARS-J 内の各順位付け手法 CR,KR,CR+KR の比較を行った．さらに，アンケートも行い使い勝手についても評価を行った．

評価の結果，SPARS-J は比較した他の検索システムとはソフトウェア検索するという点で明らかに差が見られた．また，SPARS-J 内における順位付け手法の違いにおいては，旧バージョンで比較を行った企業内ソフトウェアを用いた実験ではそれぞれの比較において，有意な差は見られなかった．しかし，新バージョンで実験したときには，ユーザとシステムの順位付けの違いの大きさの比較において 2 つの順位付けには違いが見られ，それらを組み合わせた順位付け手法においては各手法単独で順位付けを行うより順位付けの性能の向上が見られているので，評価結果は改善すると考えられる．さらに，アンケートの結果，SPARS-J は企業内ソフトウェアの保守に対して大いに役立っており，本システムはソフトウェアの開発，保守の両方において有効であると言える．

今後より詳細な評価を行う必要があり，課題として以下が挙げられる．

- 再現率の調査
- 順位付け性能以外の定量的な評価

謝辞

本研究の全過程を通して，常に適切な御指導および御助言を賜りました 大阪大学大学院 情報科学研究科 コンピュータサイエンス専攻 井上 克郎 教授に心より深く感謝致します.

本論文を作成するにあたり，逐次適切な御指導および御助言を賜りました 大阪大学大学院 情報科学研究科 コンピュータサイエンス専攻 楠本 真二 助教授に心から感謝致します.

本論文を作成するにあたり，適切な御指導，御助言を賜りました 大阪大学大学院 情報科学研究科 コンピュータサイエンス専攻 松下 誠 助手に心から感謝致します.

本研究を通して，適切な御助言を頂きました 独立行政法人 科学技術振興機構 山本 哲男 氏に深く感謝致します.

本研究を通して，適切な御助言を頂きました 大阪大学大学院 情報科学研究科 コンピュータサイエンス専攻 横森 励士 氏に深く感謝致します.

最後に，その他様々な御指導，御助言等を頂いた 大阪大学大学院 情報科学研究科 コンピュータサイエンス専攻 井上研究室の皆様に深く感謝いたします.

参考文献

[1] R. C. Seacord, S. A. Hissam, K. C. Wallnau: “Agora: A Search Engine for Software Components”, *IEEE Internet Computing*, Vol. 2, No. 6, pp. 62–70, (1998).

[2] Amanda Spink, B. J. Jansen, D. Wolfram, T. Saracevic: “From E-Sex to E-Commerce: Web Search Changes” IEEE Computer, Vol. 35, No. 3, pp. 107–109, Mar (2002).

[3] “The Apache Software Foundation”, http://www.apache.org/ .

[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] 木谷 強 ほか: “日本語情報検索システム評価用テストコレクション BMIR-J2”, 情報処理研究報告, DBS114, pp. 15–22, (1998).

[6] J. C. Borda: “M’emoire sur les ’elections au scrutin”, *Histoire de l’Acad’emie Royale des Sciences*, (1781).

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

[8] C. Braun: “NATO Standard for the Development of Reusable Software Components”, *NATO Communications and Information Systems Agency*, (1992).

[9] C. Dwork, R. Kumar, M. Naor, D. Sivakumar: “Rank Aggregation Methods for the Web”, *Proc. of WWW10*, pp. 613–622, (2001).

[10] “技術評論社”, http://www.gihyo.co.jp/ .

[11] “Google”, http://www.google.com/ .

[12] 池田 央 編: “統計ガイドブック”, 新曜社, (1984).

[13] K. Inoue, R. Yokomori, H. Fujiwara, T. Yamamoto, M. Matsushita, S. Kusumoto: “Component Rank: Relative Significance Rank for Software Component Search”, Proceedings of the 25th International Conference on Software Engineering (ICSE2003), pp. 14–24, Portland, Oregon, U.S.A., May 6-8, (2003), and Technical Report of SE Lab, Dept. of Computer Science, Osaka University, SEL-Oct-8-2002, Oct, (2002).

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

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

[16] J. Gosling, B. Joy, G. Steele, and G. Bracha: “The Java Language Specification”, Addison-Wesley, (2000).

[17] M. H. Aviram: “Code-centric search tool strives to reduce Java development time”, *Java World*, June (1998).

[18] 亀井, 門田, 松本: “WWW を対象としたソフトウェア検索エンジンの構築”, 電子情報通信学会技術研究報告, SS2002-47, Vol. 102, No. 617, pp. 59–64, (2003).

[19] 片山, 土居, 鳥居 [監訳]: “ソフトウェア工学大事典”, 朝倉書店, (1998).

[20] 北, 津田, 獅々堀: “情報検索アルゴリズム”, 共立出版, (2002).

[21] 小堀, 山本, 松下, 井上: “類似度メトリクスを用いた Java ソースコード間類似度測定ツールの試作”, 電子情報通信学会技術研究報告, SS2003-2, Vol. 103, No. 102, pp. 7–12, (2003).

[22] “Namazu Project”, http://www.namazu.org .

[23] 西, 梅森, 山本, 横森, 松下, 楠本, 井上: “Java ソフトウェア部品解析・検索システム SPARS-J の構築”, 電子情報通信学会技術研究報告, SS2003-23, Vol. 103, No. 481, pp.43–48, (2003).

[24] 庭山, 松尾, 萩原, 金田: “セマンティック Web を利用したソフトウェア検索システムの提案”, 電子情報通信学会技術研究報告, SS2002-57, Vol. 102, No. 704, pp. 27–31, (2003).

[25] NTCIR 情報検索システム評価用テストコレクション構築プロジェクト, “http://research.nii.ac.jp/ntcir/index-ja.html” .

[26] “O’reilly Media”, http://www.oreilly.com/ .

[27] J. J. Rocchio: “Performance indices for document retrieval”, In G. Salton (Ed.), The SMART retrieval system–Experiments in automatic document processing, pp. 57–67, Englewood Cliffs, NJ:Prentice-Hall, (1971).

[28] 芝, 渡部, 石塚 編: “統計用語辞典”, 新曜社, (1984).

[29] “SOURCEFORGE.net”, http://sourceforge.net/ .

[30] “Sun Microsystems”, http://www.sun.com/ .

[31] 山本, 横森, 松下, 楠本, 井上: “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”, 電子情報通信学会技術研究報告, SS2002-17, Vol. 102, No. 329, pp. 13–18, (2002).

[32] Y. Y. Yao: “Measureing Retrieval Effectiveness Based on User Preference of DocumentS”, Journal of the American Society for Information Science, Vol. 46, No. 2, pp. 133–145, (1995).

[33] 横森, 藤原, 山本, 松下, 楠本, 井上: “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”, 電子情報通信学会論文誌 D-I, Vol. J86-D-I, No.9, pp.671–681, (2003), and Technical Report of SE Lab, Dept. of Computer Science, Osaka University, SEL-Nov-21-2002, Nov (2002).

[34] 鷲崎, 深澤: “オブジェクト指向プログラムのためのコンポーネント抽出型検索システム”, オブジェクト指向 2003 シンポジウム (OO2003), (2003).