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
DE2004-111, DC2004-26(2004-10)

協調フィルタリングに基づくJavaクラスファイル推薦システム

柿元 健† 角田 雅照† 大杉 直樹† 門田 暁人† 松本 健一†
†奈良先端科学技術大学院大学情報科学研究科 〒630-0192 けいはんな学研都市
E-mail: †{takesi-k, masate-t, naoki-o, akito-m, matumoto}@is.naist.jp

あらまし 近年，多くのソフトウェア開発プラットォームでは，開発に役立つ様々なソフトウェアコンポーネントが提供されている．しかし，提供されるコンポーネントの数が膨大になり，開発者が有用なコンポーネントに気付かないという問題が生じている．本稿では，この問題を解決するため，協調フィルタリングを用いてJavaクラスファイルを推薦するシステムを提案する．開発者は，提案システムに開発中のJavaクラスファイル（開発中クラスファイル）を入力する．提案システムは開発中クラスファイル内で利用されているJavaクラスを調べ，過去に開発されたクラスファイルから似ているもの（類似クラスファイル）を探し出す．提案システムは類似クラスファイル内で利用されており，開発中クラスファイル内で利用されていないJavaクラスを開発者に推薦する．提案システムの推薦精度を4つの基準（適合率，再現率，F1値，Half-life Utility）により実験的に評価したところ，提案システムの精度が，単に良く使われるJavaクラスを推薦する場合より約4倍高くなった．

キーワード 情報フィルタリング，類似度，類似度計算，リファクタリング，J2SE

A Java Class File Recommender System Based on Collaborative Filtering

Takeshi KAKIMOTO† Masateru TSUNODA† Naoki OHSUGI† Akito MONDEN†
and Ken'ichi MATSUMOTO†

†Graduate School of Information Science, Nara Institute of Science and Technology
Kansai Science City, 630-0192 Japan
E-mail: †{takesi-k, masate-t, naoki-o, akito-m, matumoto}@is.naist.jp

Abstract Today, most software development platforms provide various software components. However, some software developers are not aware of useful components because extremely large amount of components are provided. This paper propose a system recommending the developers some Java class files by using *Collaborative Filtering*. Once a developer enters a Java class file which has been developed in ongoing project, the proposed system investigates used Java classes in the entered class file. Next, the system finds some similar class files from already completed class files which made in the past projects. Next, the system recommends the developer some Java classes used in the similar class files and not used in the entered Java class files. We experimentally evaluated recommendation accuracy of the proposed system with four criteria (*recall*, *precision*, *F1-value*, *Half-life Utility*). The results shows the proposed system outperformed quadruply the simple method recommending the most frequently used Java classes.

Keyword information filtering, similarity, similarity computation, refactoring, J2SE

1. まえがき

ソフトウェア開発者の要求や必要性を満たすべく，様々なソフトウェアコンポーネントが，多くのソフトウェア開発プラットフォームによって提供されている．提供されるコンポーネントを利用することで，開発者は多機能なソフトウェアを短期間に低コストで作成することができる[7]．例えば，Java 2 SDK, Standard Edition (J2SE) Version 1.4.1_02では基本クラスとして5568個のクラスが提供されている．

ソフトウェアを実装する際に，開発者はそれら提供されているクラスのごく一部しか利用しない．例えば，いくつかのオープンソースソフトウェアに対してソフトウェアの実装に利用されたクラスを調査したところ，実際に利用されたクラス数は平均 223.6 個であり，J2SE基本クラス全体の約4パーセントであった（図 1 を参照）．開発者が利用しなかったクラスの中には，開発者の生産性，並びに，ソフトウェアの品質を向上させる有用なクラスが含まれている可能性がある．しかし，クラスの数は膨大であるため，開発者がそれら有用なクラスを探し出すことは容易ではない，

本論文では，この問題を解決するため，協調フィルタリングを用いて有用な Java クラスを推薦するシステムを提案する．協調フィルタリングは，情報検索の分野で提案された手法であり，ユーザの好みに合うア

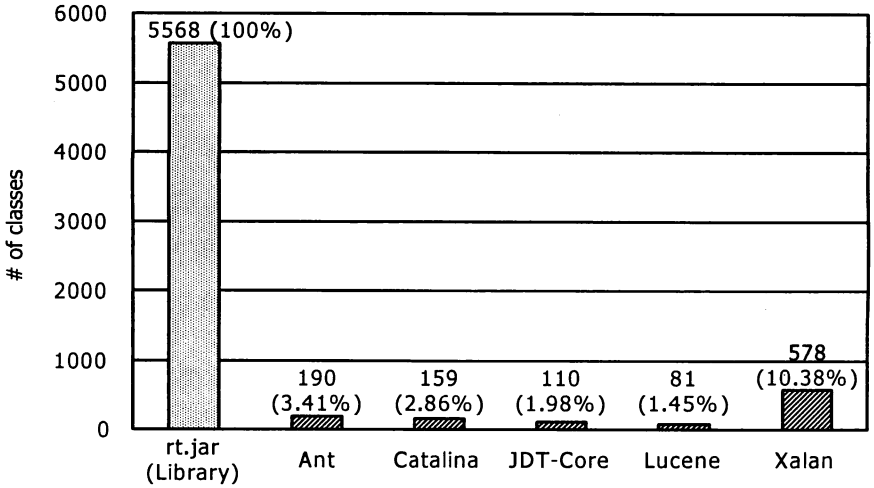


図 1 アプリケーションで使われているクラスの数

イテム（書籍，映画，音楽など）を推薦するシステムの実装に用いられる．まず，ユーザの好みに関するデータを収集する．次に，推薦対象のユーザと好みの傾向が似ているユーザ（類似ユーザ）を探し出す．最後に，類似ユーザが過去に好んで使用し，推薦対象のユーザが使用していないアイテムを推薦する．

提案システムでは，ユーザの好みに関するデータの代わりに Java クラスの使い方に関するデータを用いることで，開発中のクラスファイルの実装に役立つ Java クラスを推薦する．まず，開発中クラスファイル内の Java クラスの使い方に関するデータを収集する．次に，開発中クラスファイルと Java クラスの使い方が似ている開発済クラスファイル(類似クラスファイル)を探し出す．最後に，類似クラスファイルで使用されており，開発中クラスファイルで使用されていない Java クラスを推薦する．

以降，2 章では関連研究を紹介し，3 章では協調フィルタリングの詳細を説明する．4 章では提案システムのアーキテクチャと動作を説明し，5 章では提案システムの有効性を評価する実験について報告する．6 章で評価実験の結果について考察し，7 章でまとめと今後の課題について述べる．

2. 関連研究

市井ら[3]は Java クラス検索システム SPARS-J を提案している．SPARS-J は Java ソースコードを静的に解析して索引語を抽出し，ユーザに入力された検索語に応じて検索結果を返す．ユーザは実装したい機能などから検索語を類推し，実装に役立つ Java クラスを探し出せる．しかし，ユーザが検索語を類推できない Java クラス，並びに，ユーザが検索しようとしない（存在に気付いていない）Java クラスを探し出せない．

SPARS-J はユーザの検索履歴に基づいて協調フィルタリングを行う機能を備えている．しかし，この機能は検索効率を向上するためのものであり，前述の問題を解消するには至っていない．

Ye と Fischer [9]は，プログラムの実装に役立つソフトウェアコンポーネントを推薦する CodeBroker を提案している．CodeBroker はプログラムから抽出される特徴を用いて推薦する．また，ユーザとシステムの間のインタラクションの履歴を用いて推薦の精度の向上を図っている．しかし，推薦結果中の適切な結果の割合(適合率)は 40%以下の値しか得られていない．

3. 協調フィルタリング

協調フィルタリングは大量のアイテムの中から，ユーザの好みに合うアイテムを選び出し，推薦するシステムの実装に用いられる．一般に，協調フィルタリングを用いた推薦は次の手順で行われる．

手順1. 各ユーザは使用したアイテム(書籍・楽曲・映画など)に対する評価を行う．

手順2. 推薦対象のユーザ（対象ユーザ）と評価の傾向が似ているユーザ（類似ユーザ）を探す．

手順3. 対象ユーザが未使用のアイテムに対する評価を，類似ユーザの評価に基づいて予測する．

手順4. 対象ユーザが高く評価すると予測されたアイテムを対象ユーザに推薦する．

Rensick ら[5]は，ユーザの興味に合う Usenet ニュース記事を推薦するシステム GroupLens を開発した．また，Sarwar ら[6]は，アイテム数が非常に多く，ユーザによる評価が全体の 1%以下しか評価が行われていないデータに対して適用可能なユーザの好みを予測できるアルゴリズムを提案した．GroupLens で利用されている手法がユーザベース手法と呼ばれるのに対して，この手法はアイテムベース手法と呼ばれ，類似ユーザではなく，評価が似ているアイテム(類似アイテム)を選び出し，類似アイテムの評価を用いて推薦する．類似ユーザと比較して類似アイテムの変動は小さく，類

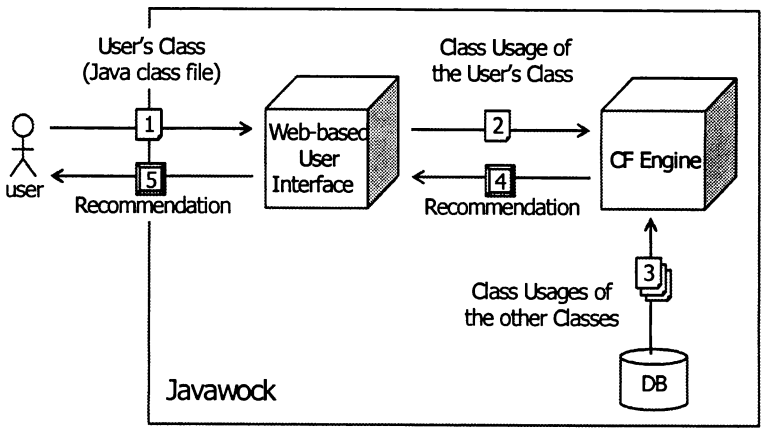


図 2 システムの構成

似アイテムのキャッシュが可能であり計算時間の短縮が行えるため，アイテム数が非常に多い場合に有効な手法である．このアルゴリズムは Amazon.com が提供する本推薦システムでも用いられている．

提案システムでは，ユーザはアプリケーション，アイテムはクラスとなり，ユーザの評価の代わりにアプリケーションがそのクラスを使用しているか，未使用であるかを用いる．また，これまでの協調フィルタリングを用いたシステムとの大きな違いは，提案システムでは使用/未使用の関係を用いるため，評価されていない値（欠損値）を扱わないことである．

4. Java クラスファイル推薦システム

4.1. システムの概要

Java クラス推薦システムは，ユーザからアップロードされた Java クラスファイルに対して推薦する．Java クラスファイルは，Java ソースファイルをコンパイルすることで作成されるバイトコードである．そのため，推薦するアプリケーションはコンパイル可能で必要がある．そこで，システムの利用対象となるユーザには，システムの振る舞いを変えずに，プログラムの冗長な部分を取り除く，柔軟性を持たせるなどのように再構築（リファクタリング）する保守作業者や，リファクタリングによる再設計を中心としてソフトウェアの開発する XP(Extreme Programming)[1]などのアジャイル開発プロセスによってコーディングする開発者などが考えられる．

Java クラス推薦システムの構成を図 2に示す．システムは，ユーザからの Java クラスファイルのアップロードを受け付け，また，ユーザへ推薦結果の提示する Web ベースのユーザインタフェース（Web-based User Interface），協調フィルタリングにより推薦を作成する協調フィルタリングエンジン（CF Engine），開発済みのアプリケーションのクラス使用/未使用のデータを蓄積しているデータベース（DB）から構成される．

システムは以下の順序で推薦を行う．

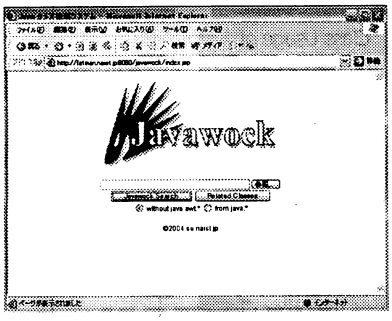
1. ユーザから Java クラスファイルが Web-based User Interface にアップロードされる（図 3-(a)）．
2. アップロードされた Java クラスファイルを解析しクラスの使用/未使用を調べる．解析には J-birth[8]の Used Classes の解析エンジンを用いた．
3. クラスの使用/未使用の解析結果を CF Engine へ渡す．
4. CF Engine は受け取った解析結果に対して，DB に蓄積されているデータを用いて協調フィルタリングによる推薦結果を作成する．
5. Web-based User Interface へ作成された推薦結果を返す．
6. Web-based User Interface は推薦結果をユーザへ提示する（図 3-(b)）．推薦結果としては，クラス名，推薦スコア，Java ドキュメントの概要とリンクをユーザへ提示する．

4.2. 推薦方法

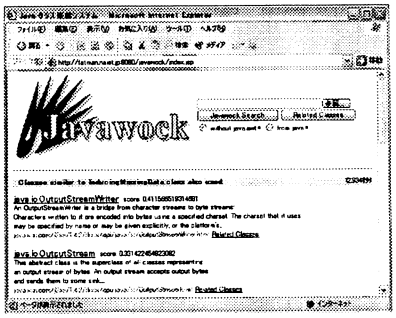
提案システムは三種類の方法で推薦する．以下でそれぞれの推薦方法について述べる．

最初の推薦方法は，協調フィルタリングのユーザベース手法を用いた，類似したクラスを使っているアプリケーションからの推薦である．この方法では，まず，ユーザからアップロードされた Java クラスファイル（ユーザクラス）と類似したクラスを使用しているアプリケーション（類似アプリケーション）を複数求める．次に，類似アプリケーションで多く使用されているクラスのうち，ユーザクラスでは未使用のクラスの推薦スコアを算出する．推薦結果として推薦スコアの高い順にユーザに提示する．

二つ目の推薦方法は，類似したクラスを使っているアプリケーションの提示である．この手法は，最初の推薦方法で，最初の推薦方法の類似したクラスを使っているアプリケーションからの推薦にの中で求めた類似アプリケーションをユーザに提示する．



(a) 入力画面



(b) 出力画面

図 3 実行画面

三つ目の推薦方法は，協調フィルタリングのアイテムベース手法を用いた，アプリケーションによる使われ方が類似しているクラスを用いた推薦である．この方法では，まず，ユーザからアップロードされたJavaクラスファイル（ユーザクラス）と類似したアプリケーションで使用されているクラス（類似クラス）を複数求める．次に，類似クラスを多く使用しているアプリケーション（類似アプリケーション）を求める．類似アプリケーションが使用しているクラスで，ユーザクラス中で未使用のクラスの推薦スコアを算出する．推薦結果として推薦スコアの高い順にユーザに提示する．

4.3. 協調フィルタリングによる推薦

協調フィルタリングエンジンにおいて実行される協調フィルタリングによる推薦は，類似度計算，推薦スコア算出の手順で行われる．それぞれの手順について以下で述べる．

類似度計算は，クラス使用/未使用のデータから類似アプリケーション，または，類似クラスを求めるために，ユーザクラスと開発済みアプリケーションの類似度を算出する．類似度計算にはCosine Similarity法[6]を用いた．ユーザクラス u と開発済みアプリケーション a の類似度 $sim(a,i)$は式(1)で求められる．ここで u_{ij} はアプリケーション i がクラス j を使用していれば1，未使用であれば0をとる変数であり．Ap_i はアプリケーション Ap の使用しているクラスの集合である．

$$sim(a,i)=\frac{\sum_{j\in Ap_a\cap Ap_i} u_{a,j}\times u_{i,j}}{\sqrt{\sum_{j\in Ap_a\cap Ap_i}(u_{a,j})^2}\sqrt{\sum_{j\in Ap_a\cap Ap_i}(u_{i,j})^2}} \quad (1)$$

推薦スコア算出は，類似アプリケーションからユーザクラスが未使用のクラスの推薦スコアを算出する．推薦スコア算出にはWeighted Sum法[6]を用いた．ユーザクラスaのクラス b の推薦スコア $R(a,b)$は式(2)で求められる．ここで，k-nearestApplication はユーザクラスとの類似度が高いk個のアプリケーションの集合である．

$$R(a.b)=\frac{\sum_{k\in k-nearestApplication} sim(a,k)\times c_{k,b}}{\sum_{k\in k-nearestApllication} sim(a,k)} \quad (2)$$

それぞれの手法は，協調フィルタリングで用いられる計算手法のうち事前実験で最も有効であった手法を用いた．

5. 評価実験

5.1. 実験に使用したデータ

実験に使用したデータには，Java 2 SDK, Standard Edition (J2SE)のバージョン1.4.1_02の基本クラスライブラリであるrt.jarに含まれるクラスのうち，主要なクラスを利用した．アプリケーションとして371個のクラスを，使用/未使用を解析するクラスとして331個のクラスを利用した．使用/未使用を解析するクラスの数が少ないのは，アプリケーションとして用いたクラスの中で全く使用されていないクラスを除外したためである．これらのクラスを用いて図4に示す $m\times n$ 行列で表されるデータセットを作成した．図中，$Ap_i\in\{Ap_1, Ap_2, ..., Ap_m\}$は i 番目のアプリケーションを表し，$c_j\in\{c_1, c_2, ..., c_n\}$は j 番目のクラスを表す．すなわち，図4の行がアプリケーション，列がクラスを表している．$u_{i,j}\in\{u_{1,1}, u_{,2}, ..., u_{m,n}\}$はアプリケーション p_i でクラス c_j を使用しているかどうかを表し，解析の結果，使用している場合は $u_{m,n}$=1，使用していない場合は $u_{m,n}$=0とした．

5.2. 実験の手順

実験は，三種類の推薦方法の中の，類似したクラスを使っているアプリケーションからの予測(EU)と，アプリケーションによる使われ方が類似しているクラスを用いた予測(EI)についての評価を行った．類似したクラスを使っているアプリケーションの提示は定量的な評価を行うことは難しく，また，類似したクラスを

	c_1	c_2	…	c_j	…	c_b	…	c_n
Ap_1	$u_{1,1}$	$u_{1,2}$	…	$u_{1,j}$	…	$u_{1,b}$	…	$U_{1,n}$
Ap_1	$u_{2,1}$	$u_{2,2}$	…	$u_{2,1}$	…	$u_{2,b}$	…	$U_{2,n}$
…	…	…		…		…		…
Ap_i	$u_{i,1}$	$u_{i,2}$	…	$u_{i,1}$	…	$u_{i,b}$	…	$u_{i,n}$
…	…	…		…		…		…
Ap_a	$u_{a,1}$	$u_{a,2}$	…	$u_{a,1}$	…	$u_{a,b}$	…	$u_{a,n}$
…	…	…		…		…		…
Ap_m	$u_{m,1}$	$u_{m,2}$	…	$u_{m,1}$	…	$u_{m,b}$	…	$u_{m,n}$

図 4　予測に用いる $m \times n$ 行列

使っているアプリケーションの予測の前段階の結果を用いるため評価実験は行っていない．実験は以下のような手順で行った．

① データセットから Ap_i を取り出し，残りのデータセットから p_i を取り除く
② Ap_i が c_j を使用しているかどうかを不明とし，$u_{i,j}$ の値の予測を行う．
③ ②を p_i の全ての $u_{i,j}$ に対して行う
④ ①～③をデータセットに含まれる全ての Ap_{ij} に対して行う

また,提案手法の性能と比較するために，平均値を用いた予測(EA)を行った．この場合の推薦スコア $R(a,b)$ は式(3)で求める．

$$R(a.b)=\frac{c_b\text{を使っているアプリケーション数}}{\text{全アプリケーション数}} \quad (3)$$

5.3. 評価基準

評価基準には適合率，再現率，F1 値，Half-life Utility の 4 つを用いた．これらは協調フィルタリングの評価基準として多く用いられている[2]．適合率は，推薦結果に含まれる適切な結果の割合を示し，再現率は，全ての適切な結果のうち推薦結果に含まれた割合を示す．F1 値は，適合率と再現率を一つの値で表現する値である．これらは推薦精度を表す評価基準で，各評価基準の値が大きいほど，予測精度が高いことを示す．Half-life Utility は，推薦スコアを用いて推薦順序を評価するための評価基準である．

適合率 ：

$$\text{適合率}=\frac{\text{予測の正答数}}{\text{使用と予測した数}}\times 100 \quad (4)$$

再現率 ：

$$\text{再現率}=\frac{\text{予測の正答数}}{\text{実際に使用している数}}\times 100 \quad (5)$$

F1 値 ：

$$F_1=\frac{2\times\text{適合率}\times\text{再現率}}{\text{適合率}+\text{再現率}} \quad (6)$$

Half-life Utility(R) :

$$R_a=\sum_j \frac{u_{a,j}}{2^{(j-1)/(\alpha-1)}} \quad (7)$$

$$R=100\times\frac{\sum_a R_a}{\sum_a R_a^{\max}} \quad (8)$$

ここで，$u_{a,j}$ はアプリケーション Ap_a に対して j 番目に推薦されたクラスの(実測)点数を表す．α は，ユーザがアイテムを確認する確率が 50%となる順位を表す．ここでは α =10 とした．R_a^{max} は推薦システムが完全な推薦を行った場合の R_a の値である．

5.4. 実験結果

式(2)の k を決定するために予備実験を行い，最も精度が高かった k=3 を採用した．EU，EI，EA それぞれの適合率と再現率の関係を表したグラフを図 5に，F1 が最も大きいときの各評価基準を表 1に示す．EU の評価基準が最も優れており，EU，EI とも EA の評価基準よりも優れていた．また，EA の適合率は最大でも 18%であった．よって，提案手法による予測は平均値による予測よりも精度が高いといえる．EU と EI の評価基準を比較すると，EU のほうが優れていた．

表 1　各予測方法における評価基準

	再現率	適合率	F1 値	Half-life Utility
EU	68%	65%	0.66	77%
EI	56%	39%	0.46	55%
EA	18%	19%	0.18	14%

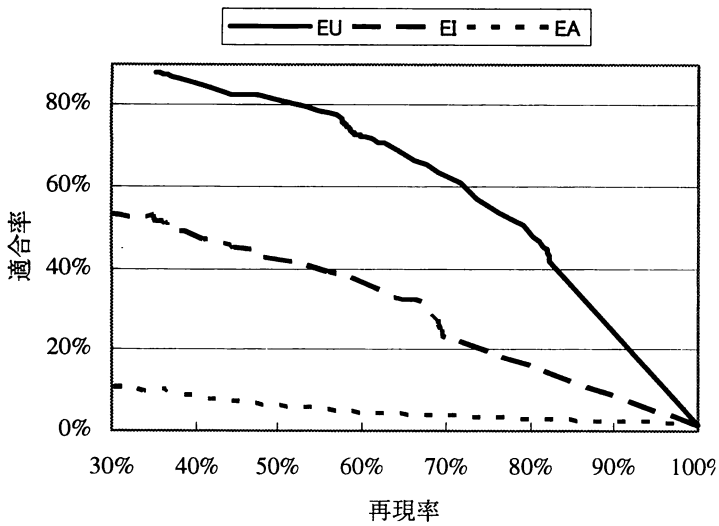


図 5　適合率と再現率の関係

6. 考察

提案システムによる推薦である EU, EI は全ての評価基準で EA よりも高い結果が得られていることから，ある開発中のアプリケーションと完成済みのアプリケーションのクラスの使い方が似ているかどうかを比較するアプローチは有効であると考えられる．ただし，EI は全ての評価基準で EU よりも低くなっていることから，この推薦方法は補完的に使うべきであると考えられる．

なお,この実験は 1 つのデータセットだけを用いたものであり,今後はさらに多くのデータセット，例えばオープンソフトウェアのコードを用いて実験し,結果の信頼性を向上させる必要があると考えられる．

また，今回の実験では，1 つのクラスの使用/未使用を不明にして予測を行ったが，例えば開発中のアプリケーションが大部分のクラスに対して使用/未使用が未確定の場合，推薦は困難であると考えられる．今後はクラスの使用/未使用の確定数と予測精度の関係を調査する必要があると考えられる．

7. むすび

本論文では，クラスの使用/未使用から協調フィルタリングを用いて Java クラスの推薦するシステムである Javawock を提案した．評価実験の結果，全ての評価基準で推薦精度が高いという結果が得られた．これより，提案手法を用いた推薦は有効であると考えられる．

今後の課題は，クラスライブラリとして提供されているクラスを用いるのではなく，実際にアプリケーションを用いて評価実験を行うことである．また，開発者に実際にシステムを利用してもらい，実用性の評価を行う必要がある．

文　献

[1] K. Beck, Extreme Programming Explained: Embrace Change, Addison-Wesley, New York, 1999.

[2] J. Herlocker, J.A. Konstan, L.G. Terveen, and J.T. Riedl, “Evaluating collaborative filtering recommender systems,” ACM Transactions on Information Systems (TOIS), Vol.22 , No.1, pp.5-53, 2004.

[3] 市井誠，山本哲男，横森励士，井上克郎，”ソフトウェア部品推薦のための協調フィルタリング手法の提案と実現”，信学技報，SS2004-15, Vol.104, No.243, pp.7-12, 2004.

[4] K. Inoue, R. Yokomori, H. Fujiwara, T. Yamamoto, M. Matsushita, S. Kusumoto, “Component Rank: Relative Significance Rank for Software Component Search,” In Proc. of the 25th International Conference on Software Engineering (ICSE2003), pp14-24, Portland, Oregon, U.S.A., May, 2003.

[5] P. Resnick, N. Iacovou, M. Suchak, P. Bergstrom, and J. Riedl, ”GroupLens: An Open Architecture for Collaborative Filtering of Netnews,” In Proc. ACM Conf. on Computer Supported Cooperative Work (CSCW’94), pp.175-186, Chapel Hill, North Carolina, U.S.A, Oct. 1994.

[6] B. M. Sarwar, G. Karypis, J. A. Konstan, and J.T. Riedl, “Item-Based Collaborative Filtering Recommendation Algorithms,” Proc. 10th International World Wide Web Conference (WWW10), pp. 285-295, Hong Kong, May, 2001.

[7] C.Szyperski, Component Software, Addison-Wesley, New York, 1998.

[8] H. Tamada, M. Nakamura, A. Monden, and K. Matsumoto, "Design and evaluation of birthmarks for detecting theft of Java programs," Proc. IASTED International Conference on Software Engineering (IASTED SE 2004), Innsbruck, Austria, Feb. 2004.

[9] Y. Ye and G. Fischer, "Supporting Reuse by Delivering Task-Relevant and Personalized Information," In Proc. of 2002 International Conference on Software Engineering (ICSE2002), Orlando, Florida, U.S.A., pp. 513-523, May, 2002.