

使用クラスに基づくソフトウェアの機能面からの分類

牛窓 朋義[†] 門田 暁人[†] 玉田 春昭[‡] 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 奈良県生駒市高山町 8916-5

[‡] 京都産業大学 コンピュータ理工学部 〒603-8555 京都府京都市北区上賀茂本山

E-mail: [†] {tomoyoshi-u, akito-m, matumoto}@is.naist.jp, [‡] tamada@cc.kyoto-su.ac.jp

あらまし ソフトウェア開発において、与えられた機能要件を実装するために、既存のクラスライブラリの中から機能の実現に役立つクラスを選択し、使用することが一般的である。このことから、ソフトウェアが使用するライブラリクラス群は、そのソフトウェアが実現する機能と強い関連を持つと考えられる。本稿では、各ソフトウェアが使用するクラスに基づいて、多数のソフトウェアを機能面から分類する方法を提案する。提案方法では、使用クラスの種類と出現頻度に基づいてソフトウェア間の距離を算出し、階層的クラスターリングによってソフトウェアを分類する。Java ソフトウェアを対象とした実験の結果、各クラスの重み付けに tf-idf 法を用い、距離の算出にユークリッド距離を用いた場合に、もっとも妥当な分類が得られた。

キーワード ソフトウェア分類, クラスライブラリ, tf-idf, ソフトウェアバースマーク

Software Classification According to their Functionality based on Used Classes

Tomoyoshi USHIMADO[†] Akito MONDEN[†] Haruaki TAMADA[‡] and Kenichi MATSUMOTO[†]

[†] Graduate School of Information Science, Nara Institute of Science and Technology

8916-5 Takayama-cho, Ikoma-shi, Nara, 630-0192 Japan

[‡] Faculty of Computer Science and Engineering, Kyoto Sangyo University

Motoyama, Kamigamo, Kita-ku, Kyoto-shi, Kyoto, 603-8555 Japan

E-mail: [†] {tomoyoshi-u, akito-m, matumoto}@is.naist.jp, [‡] tamada@cc.kyoto-su.ac.jp

Abstract In software development, to implement given functional specifications, developers generally use class files from well-known class libraries. Therefore, a set of library classes used by software seems to have a strong relationship with the functionality of the software. In this paper we propose a method to classify software from their functional viewpoints based on used classes. In the proposed method, the distance between two software is calculated based on the number of uses of each library class, and, software is classified by hierarchical clustering. As a result of an experimental evaluation, the most appropriate classification was obtained when the tf-idf method and Euclidean distance was employed to calculate the distance.

Keyword Software Classification, Class Library, tf-idf, Software Birthmark

1. はじめに

本稿では、与えられた多数のソフトウェアを機能面から自動分類(教師なし分類)することを目的とする。ここでの機能とは、ソフトウェアの主たる機能(表計算, メール送受信, メディア再生など)を指す。多数のソフトウェアを自動分類できれば、次の役に立つと考えられる。

- (1) 特定の機能のソフトウェアを探している者が、候補となるソフトウェア群を見つけるのに役立つ。
- (2) 特定のソフトウェアに満足できない者が、類似の機能を持つ他のソフトウェアを探すのに役立つ。
- (3) ソフトウェアにはどのような機能の種別があ

るかが明らかとなり、ソフトウェア分類学 (Software Taxonomy) の発展に役立つ。

- (4) SourceForge[8]などの多数ソフトウェアを蓄積するサイトにおいて、利用者のためのデータベースを構築するのに役立つ。

ソフトウェアを分類するにあたって、本稿では、各ソフトウェアが使用するライブラリクラス群に着目し、「使用クラスが似ているソフトウェアは機能も似ている」という仮定に基づいた分類を行う。各ライブラリクラスは、それぞれ何らかの機能を提供しており、それらを組み合わせることでソフトウェア全体としての機能が実現される。このことから、類似のクラス群を利用しているソフトウェアは、ソフトウェアの機能も

類似すると期待される。

提案方法では、まず、使用するクラスの種類と出現頻度に基づいてソフトウェア間の距離を算出する。次に、距離に基づいた階層的クラスタリングによって視覚的にソフトウェアの分類を行う。また、評価実験として、32 件のオープンソースソフトウェアを分類することで、提案手法の有効性を確認するとともに、今後の課題について検討する。

従来研究としては、ソフトウェアリポジトリ分類システム MUDABlue[2]が提案されているが、ソースコード中の識別子（変数名や関数名など）の類似性に基づいて分類を行うものであり、特にソフトウェアの機能の分離を目指したものではない。

一方、提案手法は、ソフトウェアの機能と結びつきが強いと考えられる「ソフトウェアが使用するライブラリクラス」のみに着目することで、低コストで、ソフトウェアの機能に即した分類を目指している。

以降、2 章で提案手法のキーアイデアについて説明し、3 章で提案手法の詳細について述べる。そして、4 章にて提案手法の有効性を検討するための実験と、その結果についての考察を述べたのち、5 章で関連研究について紹介し、6 章でまとめと今後の課題について述べる。

2. キーアイデア

ソフトウェア開発においては、与えられた機能要件を実装するために、既存のクラスライブラリの中から機能の実現に役立つクラスを選択し、使用することが一般的である。このことから、ソフトウェアが使用するライブラリクラス群は、そのソフトウェアが実現する機能と強い関連を持つと考えられる。この使用するライブラリクラス群のことを本稿では使用クラスの集合と呼ぶ。

一例として、SourceForge の Mail カテゴリに属するプロジェクトのうち、Java 言語を開発言語とするものを挙げる。これらのプロジェクトで開発されているソフトウェアには、Java 言語上からメッセージの送受信を行う機能を持ったクラスライブラリである JavaMail を用いてメッセージの送受信機能を実装したものが多数見られた。この例では、JavaMail というクラスが、メールの送受信機能の一端を担っているといえる。

提案手法では、任意の 2 つのソフトウェアの組について、使用しているクラス群が似ていれば値が小さくなり、似ていなければ値が大きくなるように距離を定義する。そして、階層的クラスタリングを用いて、距離が近いソフトウェア同士が近いクラスに配置することで、ソフトウェアの分類を行う。

なお、提案方法では、一般によく使われているクラ

スライブラリに含まれないクラス（例えば、開発者が独自に作成したクラス）は分析対象外とする。そのようなクラスは、ソフトウェアの機能を決めるのに貢献しているかも知れないが、広く使われていないために、多数のソフトウェアの機能を分類するのには役立たないと考えられるためである。

3. 提案手法

提案手法は次の手順から構成される。

Step1 ソフトウェアから使用クラスの抽出を行う

Step2 ソフトウェア間の距離行列を導出する

Step3 階層的クラスタリングにより分類する

各ステップの詳細を以降に記す。

3.1. 使用クラスの抽出

使用クラスに基づいた分類を行うために、分類対象である各ソフトウェアの使用クラスを抽出する。また、各使用クラスに対して重み付けを行うために、その出現回数を抽出する。

本稿では、JDK 6[6], Sun Microsystems[10]が提供する標準拡張ライブラリ (javax で始まるパッケージ名のもの)、Jakarta Project[5]に含まれるクラスのみを抽出対象とする。その他のクラスは広く使われていないクラスとして扱い、抽出対象に含めない。

3.2. ソフトウェア間の距離行列の導出

分類対象である各ソフトウェアの使用クラスに基づいて、各ソフトウェア同士がどの程度類似しているかを示す距離行列を導出する。

距離行列の導出にあたって、各使用クラスに対して tf-idf 法による重み付けを行う。tf-idf 法は文書中の単語出現頻度 (tf: Term Frequency) と、ある単語が文書空間中にどの程度出現するか (df: Document Frequency) の逆数 idf を用いて、それぞれの単語に対して重み付けを行う手法である。ある単語が文書空間全体の中で少数の文書にのみ出現する場合であれば、その単語を特徴的とみなして重みは重くなり、あらゆる文書に登場するような単語であれば、その単語を汎用的とみなして重みは軽くなる。

本稿では、分類対象となるソフトウェア群を文書空間、各ソフトウェアを文書、抽出した各使用クラスを単語とみなして tf-idf 法を適用する。

3.2.1. tf-idf 法による重みの導出

ソフトウェア i の使用クラス x の総出現回数を C_{ix} , ソフトウェア i の全使用クラスの出現回数の合計を S_i

とすると、使用クラス x の単語出現頻度 tf_{ix} は次の式で定義される。

$$tf_{ix} = \frac{c_{ix}}{S_i}$$

また、分析対象であるソフトウェアの数を M 、分析対象のソフトウェアのうち使用クラス x を含むソフトウェアの数を N_x としたとき、分析対象中の使用クラス x 出現割合の逆数 idf_{ix} は次の式で定義される。

$$idf_{ix} = \log \frac{M}{N_x}$$

これより、tf-idf法によるソフトウェア i の使用クラス x の重み w_{ix} は、次の式で定義される。

$$w_{ix} = tf_{ix} \times idf_{ix}$$

3.2.2. 距離行列の導出

tf-idf法を用いて導出した各使用クラスに対する重みから、ソフトウェア間の距離を求めるにはいくつかの方法が考えられる。本稿では、2つのベクトル間の距離を求める代表的な方法である、(a)ユークリッド距離、(b)重みベクトル間のなす角の2つを採用し、4章の実験において評価することとする。

(a) ユークリッド距離

ソフトウェア ij 間で共通する使用クラス y のソフトウェア i 側の重みを w_{iy} 、ソフトウェア ij 間で共通する使用クラスのソフトウェア i 側の重みの合計を $matchw_{ij}$ 、ソフトウェア i の使用クラスの重みの合計を Sw_i とすると、ソフトウェア ij 間の距離 d_{ij} は次の式で定義される。

$$d_{ij} = 1 - \frac{\frac{matchw_{ij}}{Sw_i} + \frac{matchw_{ji}}{Sw_j}}{2}$$

(b) 重みベクトルのなす角

ソフトウェア i が持つ重みベクトル $W_i = (w_{i1}, w_{i2}, \dots)$ とソフトウェア j が持つ重みベクトル W_j との内積を $\langle W_i, W_j \rangle$ とするとソフトウェアソフトウェア ij 間の距離 d_{ij} は以下の式によって定義される。

$$d_{ij} = 1 - \frac{\langle W_i, W_j \rangle}{\|W_i\| + \|W_j\|}$$

3.3. 階層的クラスタリングによる分類

階層的クラスタリングとは、多数のクラスタが用意

されたとき、それらクラスタ間の距離の近いものから順に併合し、距離を更新することを繰り返すことで、いくつかのグループにクラスタを分類する手法である。

階層的クラスタリングのアルゴリズムとしては、最短距離法や群平均法といった様々なアルゴリズムが提案されているが、本稿では群平均法を用いる。群平均法は、計算量が多いという問題点はあるが、最短距離法のように単一のクラスタが肥大化することなく、均等なクラスタの生成が期待できるためである。

4. 実験

実験では、現実のオープンソースソフトウェア群の分類を行うことで、提案手法の有効性を評価する。評価にあたっては、tf-idf法による重み付けの効果を明らかにするために、重み付けを行った場合と行わなかった場合の両方について評価する。また、距離行列の算出にユークリッド距離を用いた場合とベクトルのなす角を用いた場合についても比較を行う。

4.1. 実験環境

本稿では、ソフトウェアから使用クラスを抽出するために、ソフトウェアバースマーク抽出ツールであるStigmata[9]を用いた。

また、距離行列を入力としたクラスタリング、およびデンドログラムの生成のために、統計解析システムR 2.9.0[7]を用いた。

4.2. 実験対象

本実験の対象として、オープンソースソフトウェア開発リポジトリであるSourceForgeに登録されているプロジェクトの中から、Java言語を開発言語とするプロジェクトを6つのドメインから検索し、検索結果の上位のうち、プラグインやフレームワークを除くソフトウェアがダウンロード可能な状態で提供されているプロジェクト32件を分類対象とした。

32件のプロジェクトのドメインと件数の内訳は次の通りである。各ドメインは、ソフトウェアの主な機能を表していると考えられる。

表1. 分類対象ソフトウェアの内訳

ドメイン	件数
3D First Person Shooters	5
Calendar	5
Chemistry	5
Internet Relay Chat	6
Mail	6
SUDOKU	5

4.3. 実験手順

本実験は以下の手順で行った。

- (1) Stigmata を用いて、実験対象の各ソフトウェアから使用クラスとその出現回数を抽出する。
- (2) 各ソフトウェア間の距離行列を、tf-idf による重み付けを行い、ユークリッド距離またはベクトルのなす角を用いる場合、および重み付けを行わず、ユークリッド距離を用いる場合の3通り導出する。

重みなしユークリッド距離は、ソフトウェア i が持つ使用クラスの種類の数を UC_i 、ソフトウェア ij 間で共通する使用クラスの種類の数を $match_{ij}$ とするとき、ソフトウェア ij 間の距離 d_{ij} は以下の式によって定義される。

$$d_{ij} = 1 - \frac{2 \cdot match_{ij}}{UC_i + UC_j}$$

- (3) 導出した距離行列を入力として、R 2.9.0 を用いて群平均法で階層的クラスタリングを行ったのち、その結果からデンドログラムを生成することで、各ソフトウェアの分類結果を視覚的に確認する。デンドログラムとは、各クラス同士がクラスタリングによって併合されていく様を表わした二分木のことであり、横軸はクラス同士の距離を示している。

4.4. 実験結果

クラスタリングの結果から生成したデンドログラムを図 1 に示す。図中の Chem は Chemistry, IRC は Internet Relay Chat, Sudk は SUDOKU を意味する。

図 2 は、クラスタ間の距離 0.5 を上限としてグループ化を行ったものである。図 2 における得られたグル

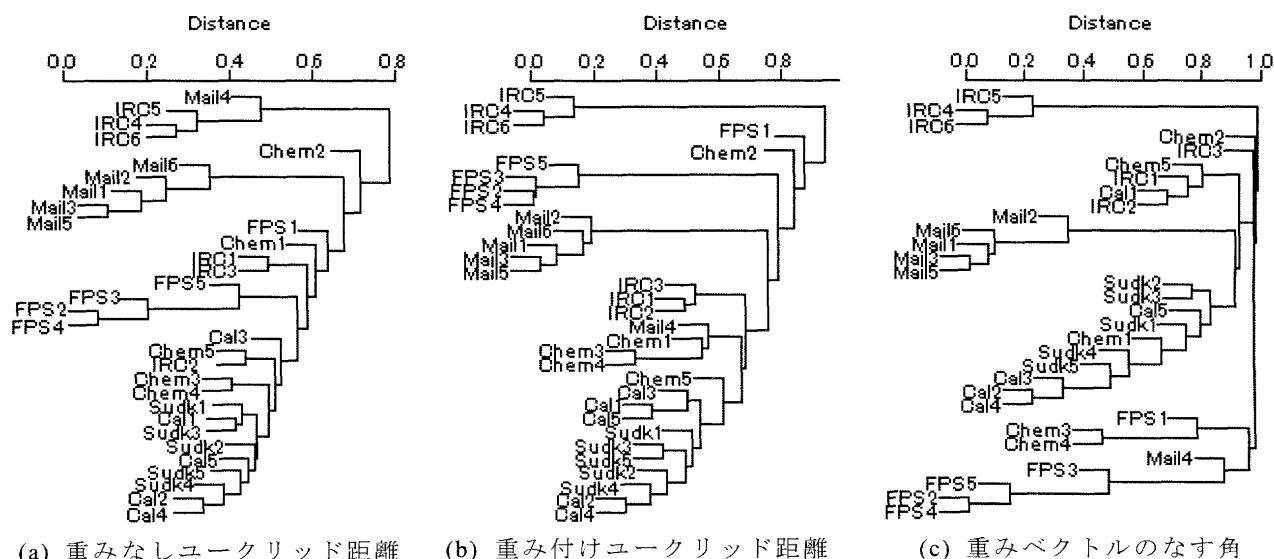


図 1. (a), (b), (c) のデンドログラムの出力結果

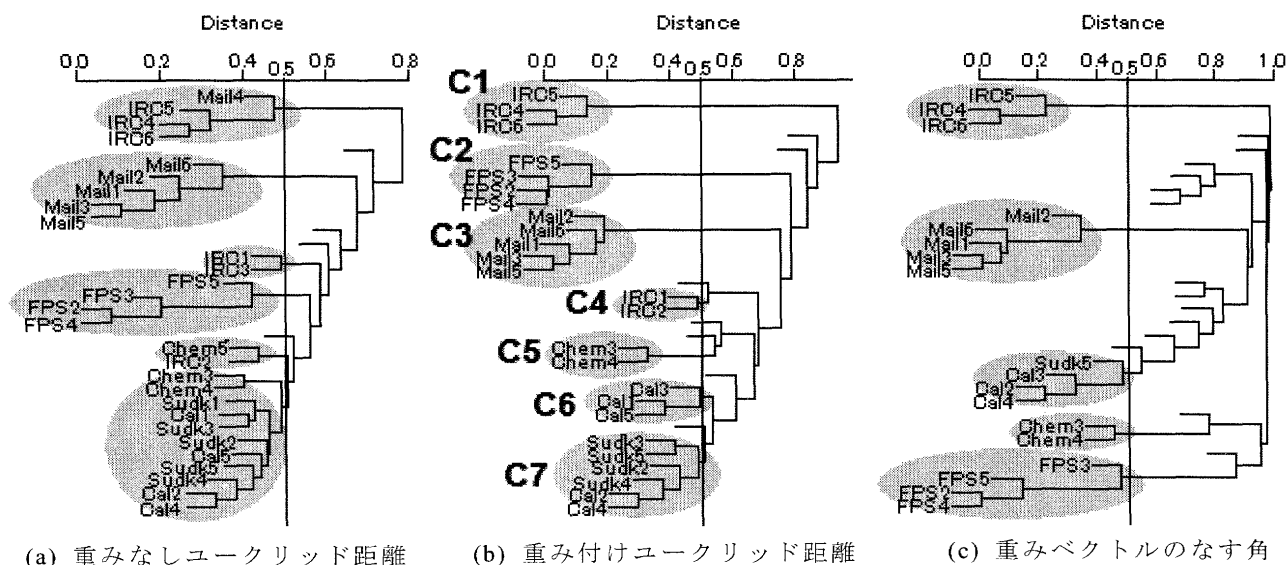


図 2. (a), (b), (c) の距離 0.5 以下のクラスタ集合をグループ化したデンドログラム

ープと、表1のソフトウェアの分類を比較した結果、重みなしのユークリッド距離による分類手法における妥当な分類件数は14件、重み付けユークリッド距離では同23件、重みベクトルのなす角では同17件であった。このことから、tf-idf法による重み付けを行い、ユークリッド距離を用いた場合において、最も高い精度で分類ができたと言える。

4.5. 考察

tf-idf法による重み付けを行い、ユークリッド距離を用いた場合の結果である図2(b)の7つのクラスタグループC1～C7について、以下に考察を述べる。

4.5.1. 妥当な分類結果を得たもの

グループC1, C2, C3については、いずれの手法でも他のクラスタと0.5以上の距離を持ったクラスタグループを形成しており、妥当な分類結果であると考えられる。その要因として、これらのクラスタグループ内でのみ共通する使用クラスが多数認められたことが挙げられる。それらのクラスのうち、重み付けを行ったときに高い重みを得たクラスの一部を表2に示す。

表2. ドメイン内での頻出使用クラス

クラスタ	使用クラス名
C1	javax.microedition.lcdui.Command
	javax.microedition.lcdui.Display
	javax.microedition.lcdui.TextField
C2	java.nio.Buffer
	java.nio.IntBuffer
	java.lang.StringBuffer
C3	java.lang.StringBuilder
	com.sun.mail.imap.IMAPFolder
	javax.mail.Folder

4.5.2. 重み付けにより精度が向上したもの

C4, C5, C6については、重みなしの場合では広く分散して分類に失敗しているが、重み付けユークリッド距離の場合では、結果に改善が見られた。

C4については、Javaの標準GUIライブラリとして提供されているjava.awtパッケージに含まれるクラスが多数共通していたことが、ソフトウェア間距離の短縮に繋がったものと考えられる。

C5についても同様で、ベクトル・数列演算を行うためのパッケージであるjavax.vecmathに含まれる使用クラスが共通していたためと考えられる。

C6については、共通する使用クラスのほとんどがjava.awtやjavax.swingなどのGUI構築用パッケージに含まれるクラスの一致が他のソフトウェアよりも高い割合で見られたことが理由である。

4.5.3. 分類に失敗したもの

C7については、重みなしの場合ではすべてが0.5程度の距離に集中して分類に失敗し、重み付けユークリッド距離の場合でも分類には至らなかった。分類が失敗した理由として、これらのドメインの多くが、汎用性の高いクラスばかりを使用しており、ドメイン単位での特徴がなく、ドメイン間で距離の違いが生じなかったためであると考えられる。

また、いずれのクラスタグループにも属さなかったソフトウェアは、そのソフトウェアにしか出現しない使用クラスを多数含んでおり、他のソフトウェアとの距離の値が大きくなってしまったものと考えられる。

4.5.4. プラットフォームの影響

IRC4,5,6およびMail4はいずれも対象のプラットフォームとして携帯電話等のモバイル環境を想定している。そのため、これらのソフトウェアには、モバイル環境向けのパッケージであるjavax.microeditionライブラリに含まれるクラスの使用が多数確認された。IRC関連のソフトウェアがC1とC4とに分かれたのは、このことが原因であると考えられる。

5. 関連研究

5.1. ソフトウェアの分類

川口らは大規模ソフトウェアリポジトリを対象とした自動分類システムMUDABlueを提案している[2]。MUDABlueは各ソフトウェアのソースコード中の識別子（変数名や関数名など）の情報を入力とし、LSA(Latent Semantic Analysis)を用いてカテゴリ分類の作成とソフトウェアの分類を行う。MUDABlueは、特にソフトウェアの機能の分類を目指したものではなく、識別子の類似する多様なカテゴリ（用途、使用ライブラリ、アーキテクチャなどによる分類）を得ることができる。一方、提案手法では、ソフトウェアの機能と結びつきが強いと考えられる「ソフトウェアが使用するライブラリクラス」のみに着目することで、ソフトウェアの機能に即した分類を目指している。提案方法は、ライブラリクラスの使用情報をだけを使っているため、スケーラブルであり、大量のソフトウェアを分類するのに向いている。また、変数名などの開発者が独自に定義する情報を用いていないため、人的要因に結果が影響されにくいと考える。

5.2. ソフトウェアバースマーク

ソフトウェアの分類を行うためには、ソフトウェアから何らかの特徴量を抽出し、それらに基づいてソフトウェア間の距離や類似度を定義する必要がある。ソフトウェアバースマーク[1][3][4][12]とは、そのような

特微量の一種であり，本稿で用いたライブラリの「使用クラス」もバースマークの一種である．

我々のグループでは，ソフトウェアバースマークを用いてソフトウェアモジュールの分類を試みたことがある[11]．ただし，分類のために用いたバースマークは，使用クラスだけでなく，変数の初期値，クラスの継承関係，メソッドの呼び出し系列を加えた4種類であり，機能による分類が必ずしも行えなかった．本研究では，過去の研究を発展させ，分類のために用いる情報を「使用クラス」に限定することで，ソフトウェアの機能面からの分類を目指している．

6. まとめと今後の課題

本稿は，使用クラスの情報を用いて，多数のプログラムを機能面から分類する手法の提案を行い，実験として6ドメイン計32件のソフトウェアを分類することで，提案手法の有効性を評価した．実験の結果，提案手法ではユークリッド距離を用いる場合に，最も妥当な分類結果が得られた．

今後さらに実験を拡張し，異なるドメインのソフトウェアを用いた実験や，分類対象となるソフトウェア数を増やした実験，単一のドメインに属するソフトウェアを分類する実験などを行うことで，提案手法の有効性の確認，および分類精度の向上に取り組む予定である．

謝辞

本研究の一部は，文部科学省「次世代IT基盤構築のための研究開発」の委託に基づいて行われた．また，本研究の一部は，文部科学省科学研究補助金(基盤研究(B)，課題番号:17300007)の補助を受けた．

文 献

- [1] 岡本 圭司，玉田 春昭，中村 匡秀，門田 暁人，松本 健一，“API 呼び出しを用いた動的バースマーク,” 電子情報通信学会論文誌 D, Vol.J89-D, No.8, pp.1751--1763, August 2006.
- [2] 川口 真司，パンカジ ガーグ，松下 誠，井上 克郎，“MUDABlue: ソフトウェアリポジトリ自動分類システム,” 電子情報通信学会論文誌 Vol.J88-D-I, No. 8, pp.1217--1227, August 2005.
- [3] Haruaki Tamada, Masahide Nakamura, Akito Monden, Ken-ichi Matsumoto, “Java Birthmarks --Detecting the Software Theft—,” IEICE Transactions on Information and Systems, Vol. E88-D, No. 9, pp. 2148--2158, September 2005.
- [4] Hyun-il Lim, Heewan Park, Seokwoo Choi, Taisook Han, “Detecting Theft of Java Applications via a Static Birthmark Based on Weighted Stack Patterns,” IEICE Transactions on Information and Systems, Vol.E91-D No.9 pp.2323-2332, September 2008.
- [5] Jakarta Project, “The Jakarta Site - The Jakarta Project -- Java Related Products,” (online), available from <<http://jakarta.apache.org/>>.
- [6] JDK, “Java SE ダウンロード -Sun Developer Network (SDN),” (online) available from <<http://java.sun.com/javase/ja/6/download.html>>.
- [7] R, “The R Project for Statistical Computing,” (online) available from <<http://www.r-project.org/>>.
- [8] SourceForge, “SourceForge.net: Find and Develop Open-Source Software,” (online) available from <<http://sourceforge.net/>>.
- [9] Stigmata, “Stigmata-Java birthmark toolkit,” (online) available from <<http://stigmata.sourceforge.jp/>>.
- [10] Sun Microsystems, “Sun Microsystems,” (online) available from <<http://www.sun.com/>>.
- [11] Takeshi Kakimoto, Akito Monden, Yasutaka Kamei, Haruaki Tamada, Masateru Tsunoda, and Ken-ichi Matsumoto, “Using Software Birthmarks to Identify Similar Classes and Major Functionalities,” In Proceedings of the 3rd International Workshop on Mining Software Repositories (MSR2006) Mining Challenge 2006, pp.171-172, May 2006. (Shanghai, China)
- [12] 玉田 春昭，森崎 修司，門田 暁人，松本 健一，“バースマークを用いたソフトウェア拡張開発での流用度合いの測定,” 情報処理学会第70回全国大会, pp.5-399--400, March 2008.