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.

名前の重複を考慮したJavaソフトウェア部品間の利用関係解析手法の提案

市井　誠[†]　横森　励士[††]　井上　克郎[†]

† 大阪大学 大学院情報科学研究科
†† 南山大学 数理情報学部 情報通信学科
E-mail: †{m-itii,inoue}@ist.osaka-u.ac.jp, ††yokomori@it.nanzan-u.ac.jp

あらまし　近年のオープンソースソフトウェア開発の活発化により，ソフトウェア部品検索システム（ソースコード検索システム）を用いたソースコード単位のソフトウェア再利用が頻繁に行われてきている．再利用の為には部品の理解が必要であり，部品を解析して得られる部品間の利用関係は再利用を効率化する為の重要な情報である．しかし，既存のソフトウェア部品検索システムでは，部品間の利用関係を得られない，もしくは部品名の重複を考慮していないために部分的に得ることができるのみである．本稿では，名前の重複を考慮した，部品間の利用関係の解析手法を提案する．提案手法では，検索システムの利用者が適切に利用関係を扱えるように，部品間の利用関係として属性付きの関連を解析する．また，提案手法に基づき，オープンソースソフトウェアのリポジトリの部品を利用する部品検索システムを提案する．さらに，試作システムを用いて適用実験を行い，提案手法の有効性を検証する．
キーワード　ソフトウェア再利用，ソフトウェア部品検索，利用関係，ソフトウェアリポジトリ

An Approach to analyze use-relation between software components with duplicated names

Makoto ICHII[†], Reishi YOKOMORI[††], and Katsuro INOUE[†]

† Graduate School of Information Science and Technology, Osaka University
†† Department of Information and Telecommunication Engineering, Nanzan University
E-mail: †{m-itii,inoue}@ist.osaka-u.ac.jp, ††yokomori@it.nanzan-u.ac.jp

Abstract　Source code reuse becomes popular using software component retrieval system (source code search system) because of the recent growth of open source software development. Use-relation between components, which is acquired by analysis of source code, is effective information for making reuse efficient. However, existing retrieval systems analyze use-relation between components insufficiently. Therefore, we propose use-relation analysis method considering name duplication. In our model, each use-relation has attributes, by which developers can choose proper components. In addition, we propose brand-new software component system using open source software repository based on proposal use-relation analysis method.
Key words　Software reuse, Software component retrieval, Use-relation, Software repository

1. はじめに

ソフトウェア再利用はソフトウェア開発の効率化に有効であるとされている．従来，再利用は再利用可能なソフトウェア部品（部品）を計画的に開発することで実践されてきたが [1]，近年のオープンソースソフトウェア開発の活発化に伴い，開発者の必要に応じて部品を検索および取得する軽量な再利用 [2] が注目されてきている．オープンソースソフトウェアの開発は公開されたソフトウェアリポジトリ（版管理システム）を用いて行われることが多く，過去のソースコードや，まだリリースされていないソフトウェアのソースコードも取得できる．

部品を再利用する為には，まず，再利用可能な部品を検索して取得する必要がある．また，開発しているソフトウェアに取得した部品を組み込む為には，その部品が依存する他の部品の取得や利用方法の理解が必要となる．しかし，大量のソフトウェアの中から必要な部品を見つけることは困難であり，また，

一般に部品の文書は不十分であることが多い．再利用の効果を向上させるためには，このような再利用にかかる労力を削減することが必要である．

SPARS-J [3]〜[5] や Koders [6] などのソフトウェア部品検索システム（ソースコード検索システム）は，オープンソースソフトウェアとして公開されているソフトウェアを検索するためのシステムである．これらのシステムを用いることで，ソフトウェアリポジトリ中の部品を検索することが出来る．しかし，検索可能な部品は，最新版などの，ソフトウェアリポジトリ中の特定のスナップショットに含まれるもののみである．そのため，検索された部品に不具合が含まれるなどの理由により，同じ部品の他のバージョンが必要となる場合，開発者は他の手段を用いて部品を取得する必要がある．また，機能変更などにより削除された部品は検索出来ない．

また，依存関係や利用法の理解の支援する情報として，部品間の利用関係が挙げられる．ある部品が利用する部品を同時に取得することで依存関係を解決でき，また，利用される部品のソースコードは部品の利用例として理解に用いることが出来る．しかし，既存のシステムは，SPARS-J を除き，利用関係を提供していない．また，SPARS-J における利用関係は，同じ名前の部品の存在を考慮していないために不完全である．同じ部品の異なるバージョンが存在する場合など，同じ名前の部品が複数存在する場合には，それら全ての可能性を考慮して利用関係を解析する必要があるが，SPARS-J ではいずれか 1 個のみを用いて利用関係を構築している．そのため，利用関係を辿って得られる情報には，不必要な情報や誤った情報が含まれており，検索の効率を下げる原因となっている．

本稿では，名前の重複を考慮した利用関係解析手法を提案する．提案手法では，開発者が利用関係を選択出来るように，同じ名前をもつ候補全てを用いて利用関係を構築する．さらに，開発者が利用関係を辿る際に適切に必要なものを選択できるように，構築された利用関係に重要度などの属性を与える．また，提案する利用関係解析手法に基づくソフトウェア部品検索システムを提案する．提案するシステムは，ソフトウェアリポジトリ中の部品の複数の版を索引付けし，部品間の利用関係を解析する．さらに，提案する利用関係解析手法を実装した試作システムおよびそれを用いた適用実験について述べる．

以降，2. では利用関係解析の問題および提案手法について述べ，3. ではソフトウェアリポジトリに基づく部品検索システムについて述べる．4. では試作システムおよび適用実験について述べる．5. では関連研究について述べ，最後に 6. でまとめと今後の課題について述べる．

2. 利用関係解析手法

2.1 基本となるモデル

提案手法の説明に先駆けて，基本となるモデルについて説明する．このモデルは，SPARS-J の利用関係解析において用いられているモデルである．まず，いくつかの概念を定義する．

- **エンティティ (entity)**: 部品から利用される要素．部品そのものもエンティティである．例えば Java の場合，クラス（インターフェースなどを含む）およびメソッド，フィールドがエンティティとなる．エンティティは名前を持ち，名前により参照される．
- **参照 (reference)**: エンティティに含まれる，他のエンティティを利用する記述．例えば，変数宣言における型や，メソッド呼び出しである．なお，簡単の為，説明中では参照同士は依存関係を持たないものとする．例えば，foo.bar().baz() のようなメソッドのカスケード呼び出しでは，foo, bar, baz それぞれが参照であり，bar は foo の指し示すエンティティに依存し，baz は bar に依存する．
- **インディケーション (indication)**: 参照とエンティティの組．参照によりエンティティが利用されることを示す．

基本となるモデルにおける，これらの概念の関連を UML 風の記法を用いて図 2 に示す．

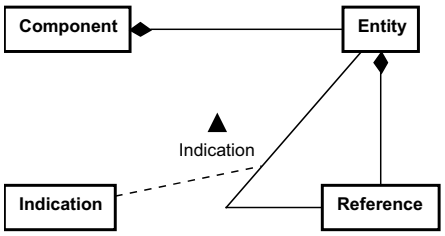

図 1 基本となるモデル

図 2 Basic Model

参照が指し示す先は，参照の持つ名前，参照を含むエンティティから利用可能なエンティティ集合およびそれらの名前を用いて特定される．基本となるモデルにおいて参照とエンティティは多対一の関連である．つまり，エンティティがその名前により一意に特定されることが前提となっている．

F を参照の集合, E をエンティティの集合とし $f \in F$, $e \in E$ とする．また，エンティティ e_1 がエンティティ e_2 を利用する利用関係を r_{e_1,e_2} で表す．利用関係を解析する問題とは，f が e を指し示すことを表す $i_{f,e}$ の集合 I を求める問題である．参照 f_1 がエンティティ e_1 に含まれ，f_1 が e_2 を指し示す，つまり i_{f_1,e_2} があるとき，r_{e_1,e_2} が導出される．

2.2 名前の重複

基本となるモデルでは，エンティティが名前により一意に特定されることを前提としているが，一般には一意に特定することは出来ないことが多い．単一のソフトウェアなどのビルド単位となる部品集合解析する場合には名前が重複することは少ないと考えられるが，本稿で対象とする，ソフトウェア部品検索システムのもつ部品集合を解析する場合には名前が重複を考慮する必要がある．名前の重複は以下の 4 通りに分類される

- **異なる版の部品**: 同じ部品の異なる版が複数含まれる場合．ソフトウェアリポジトリから複数の版の部品を収集する場合は，この種類の重複が多く発生すると考えられる．
- **API の実装**: 同一の API が，複数の実装をもつ場合．Eclipse SWT のようにプラットフォーム毎に実装される場合や，Java SE のような標準 API が複数のベンダにより実装される場合が存在する．

- **コピー**: 同一の部品が，コピーされて利用されている場合．完全に同一ではなく，変更されていることもある．
- **偶然**: 無関係な部品が，偶然に同じ名前をもつ場合．単純な名前の場合に発生することがある．

名前の重複は，ソフトウェアリポジトリから複数の版を取得しない場合，つまり特定のスナップショットや，アーカイブファイルからの収集のみであっても考慮する必要がある．demo.spars.info の部品データベース(注1) に含まれる 302,545 個の Java 部品（クラス）を調査したところ，部品集合は名前（完全限定名）に基づき 249,552 個のグループに分類され，そのうちの 34,642 個のグループは 1 個以上の部品を含み，207 個のグループは 10 個以上の部品を含むことが明らかになった．例えば，`org.w3m.dom.Document` という名前の部品は 3 つ存在する．また，そのうちの 1 個は他の 2 個よりも多数のメソッド定義を持ち，異なる版であると推測される．

参照の指し示す部品として選択すべきものは開発者の目的によって異なり，検索システム側で 1 個を選択すべきではないと考えられる．その理由として，ソフトウェアリポジトリから取得した部品は，特に最新のスナップショットの場合，十分にテストされていない可能性があり別のリビジョンの利用を考慮する必要性が考えられる為である．逆に，最新版を避けて安定版を収集した場合も，本来の開発者にとって十分安定している部品が，再利用者にとっては意図しない挙動をもつことも考えられる．さらに，安定した部品を望む開発者もいれば，新しい部品を望む開発者もいるため，利用関係の解析では候補を絞るべきでは無いと考えられる．

SPARS-J の利用関係解析は，2.1 に述べたとおり名前の重複を考慮しないモデルに基づいている．しかし，実際に解析対象である部品集合は名前の重複を含むため，1 個の参照 f に対して複数のインディケーションの候補集合 I が存在する場合には，インディケーション $i_{f,e} \in I$ を 1 個選択する．この選択は，f を含む部品と e を含む部品のファイルパスに基づいている．この方法は，収集したソフトウェアをファイルシステム上に展開する方法に依存するため，適切でない候補を選択する可能性がある．また，既に述べた通り，候補を絞り込むこと自体が適切ではない．そのため，SPARS-J では利用関係を示す際には同じ名前をもつ部品を同時に示している (図 3)．しかし，全ての部品が区別無く示されるため，開発者は，部品を選択するためにそれぞれの部品を全て調査する労力が必要となる問題がある．

2.3 提案モデル

本節では名前の重複を考慮したモデルについて説明する．提案するモデルは，2.1 節で説明したモデルの拡張である．

まず，名前の重複する複数の部品に対する利用関係を扱うために，参照とエンティティの関連を多対多で扱う．つまり，1 個の参照に複数のインディケーションが対応することがある．

また，利用関係の絞り込みを支援するため，インディケーション $i_{f,e}$ は属性 D を持つ．D は，名前と部品の対応 $d = (n, e_c, c, p)$ の集合である．ここで，n は部品の名前，e_c はエンティティとしての部品，c，p はそれぞれその対応の**確信度** (confidence) および **優先度** (priority) である． 確信度 c は，$C = \{\mathrm{DEF}, \mathrm{ALT}\}$ の要素である．優先度はディレクトリ構成などをもとに求められる値であり，同じ確信度をもつインディケーションを比較するために与える．DEF, ALT はそれぞれ名前と部品の対応が確実であるか代替的であるかを表す．名前と部品の対応に関して，その確からしさに対して何らかの根拠が存在するとき，その対応は DEF となる．さもなくば ALT となる．参照を含む部品を e_{src} とするとき，名前 n と部品 e_c の対応が DEF である為の条件は以下の通りである．

- e_{src} と e_c の出自は同一であり，版管理されていない．
- e_{src} と e_c の出自が同一のリポジトリであり，かつ，それぞれの存在期間が重複する．

上記において，出自とはリポジトリやアーカイブファイルなど，部品の収集元のことを指す．また，存在期間とは，版管理されている部品の，対象のリビジョンがコミットされてから，次のリビジョンがコミットされるまでの期間である．また，優先度をファイルシステム上での近いものから順に高い値を与える．優先度の値は，0 が最も高い優先度を表し，値が大きくなるほど低い優先度を表す．優先度は，確信度が DEF であるインディケーションは，確信度が ALT であるインディケーションよりも高い値を持つように与える．インディケーションの確信度および優先度は，それぞれ，属性に含まれる確信度および優先度のうち最も低いものが用いられる．

図 3　SPARS-J における同名部品の表示
Fig. 3　Displaying components with a same name by SPARS-J

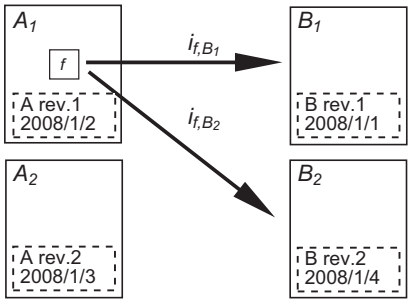

図 4　利用関係解析の例
Fig. 4　Example of Use-Relation Analysis

(注1)：2008 年 1 月現在

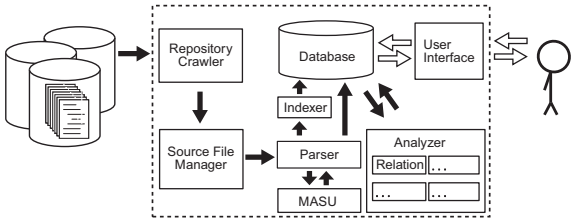

図 5　提案するシステム
Fig. 5　Proposal system

例を図 4 に示す．A_1 および A_2 はそれぞれ名前 “A” をもち，A_2 は A_1 の次のリビジョンである．同様に，B_1 および B_2 はそれぞれ名前 “B” をもち，B_2 は B_1 の次のリビジョンである．A_1 は “B” を利用する参照 f を含み，インディケーション i_{f,B_1} および i_{f,B_2} が生成される．A_1 と B_1 は存在期間が重複するため，i_{f,B_1} は属性 {(“B”, B_1, DEF, 0)} をもつ．一方，A_1 と B_2 は存在期間が重複しないため，i_{f,B_1} の属性は {(“B”, B_2, ALT, 1)} となる．

3. ソフトウェアリポジトリを利用した部品検索システム

3.1 概　　要

本稿で提案する利用関係解析を実装するソフトウェア部品検索システム SPARS 2 （仮）について説明する． 提案するシステムは，SPARS-J と同様にソースファイルの全文検索システムであるが，部品をリポジトリから収集するサブシステムを持ち，名前の重複に対応した利用関係解析を実装することで，再利用支援の観点からより有用性の高いシステムを目標とする．

3.2 システム構成

提案するシステムの構成を図 5 に示す．

- **Repository Crawler**: オープンソースソフトウェアのリポジトリから，ソースファイルを収集する．最新版などの特定のスナップショットではなく，全ての版を取得する．
- **Source File Manager**: Repository Crawler により収集されたソースファイル集合をファイルシステム上に展開し，その集合を Parser に渡すことでデータベースへ登録する．
- **Parser**: ファイルをパースして部品情報を抽出し，データベースへ登録する．MASU [7] を利用することで，複数の言語に対応する．
- **Analyer**: 様々な解析をおこなう．本稿で提案する利用関係解析はそのひとつである．他に，SPARS-J に実装されている 類似部品の解析や Component Rank の計算をおこなう．
- **Indexer**: 全文検索の為の索引付けをおこなう
- **User Interface**: 開発者が Web ブラウザを用いて部品検索を行うためのインターフェース．ソースコードの全文検索および利用関係の閲覧機能を提供する．
- **Database**: 部品情報や利用関係など，部品集合に関するデータを格納する．

4. 適 用 実 験

提案手法が大規模な部品集合に対して適用可能であるかどうか，また，従来のモデル（SPARS-J のモデル）における問題が解決しているかどうかを示すため，提案手法を実装したシステムを試作し，適用実験を行った．

4.1 実装および環境

試作したシステムは，Java を対象とし，3. 節で述べた構成要素のうち Database, Parser, Relation Analyzer および User Interface の一部を実装している．UI の機能としては，クラス名による検索とソースコードの表示，正方向および逆方向の利用関係の閲覧機能を実装した．また，出自と確信度，優先度を用い，表示される利用関係を絞り込む機能を実装した．

4.2 実 験 内 容

まず，提案手法を以下の 2 つのデータセットに対して適用し，解析された部品や参照，インディケーションの個数，解析に要した時間を示す．

- **D1** JDK 1.2.2 および JDK 1.4.2
- **D2** D1 に 91 個のオープンソースソフトウェアを加えたもの．Eclipse 2.1.3/3.0.1, NetBeans 3.6, JBoss 3.2.5, Apache Ant 1.6.2, Tomcat 3.3.2/4.1.30/5.0.28/5.5.2, Struts 1.2.4, Commons-net 1.2.2 など．

実験環境を 1 に示す．なお，確信度を求めるための出自は，それぞれのソフトウェアの配布パッケージと対応させて登録を行った．また，D2 を用いて構築したデータベースを用い，従来の利用関係のモデルにおける問題が解決されることを示す．

4.3 結　　果

D1 および D2 を解析した結果を表 2 に示す．解析時間は，ソースコードをパースして部品情報をデータベースに登録する段階と，登録された部品間の利用関係を解析する段階に分けて示されている．

a） 利 用 例 1

実装したシステムのユーザインタフェースのスクリーンショットを図 6 に示す．これは，JDK 1.2.2 に含まれる部品

表 1　実 験 環 境
Table 1　Environment of experiments

CPU	Opteron252 x2
RAM	16GB (8GB for JVM)
OS	Linux 2.6.23
JVM	1.6.0
DBMS	H2 Database Engine

表 2　解析結果概要
Table 2　Summary of analysis result

	部品数	参照数	インディケーション数	
			すべて	優先度: 0
D1	19,963	1,671,609	1,137,294	764,117
D2	127,903	11,539,473	10,543,794	5,534,129

	解析時間 (分)	
	Parser	Analyzer
D1	14	7
D2	110	211

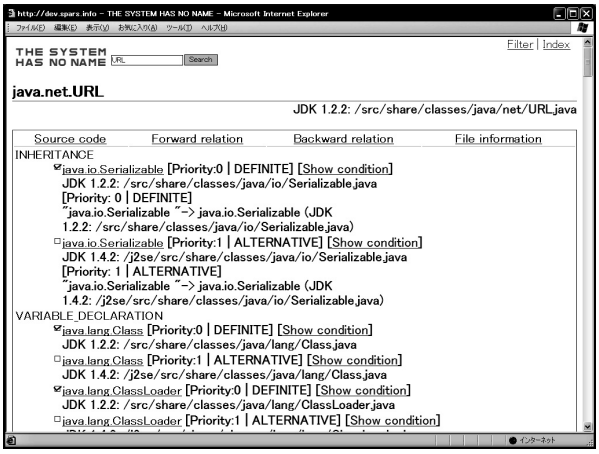

図 6　スクリーンショット: 利用関係 (1)
Fig. 6　Screenshot: use-relation (1)

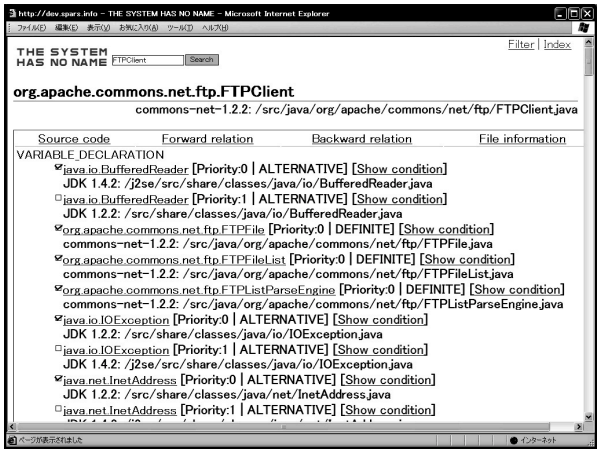

図 7　スクリーンショット: 利用関係 (2)
Fig. 7　Screenshot: use-relation (2)

java.net.URL の利用関係（利用する部品）の一覧を表示した画面である．利用する部品として，JDK 1.2.2 に含まれる部品および JDK 1.4.2 に含まれる部品が同時に表示されている．これらのうち，JDK 1.2.2 に含まれる部品には DEF (DEFINITE) が示されており，また，優先度も最も高い値である 0 が示されている．画面上では，アイテマイズにチェック入りのマーカーを用いることで優先度が最も高い部品を示している．これより，システムを利用する開発者に対し，現在注目している部品である JDK 1.2.2 の部品と同じ出自の部品が利用関係として適していることを示している．SPARS-J では，同様の状況において，それぞれの部品が同列で示されるため，開発者は手作業で適切な部品を選択する必要がある．

b）利用例 2

図 7 に，Commons-net の部品である org.apache.net.ftp.FTPClient の利用関係を示す．利用する部品として，確信度が DEF である commons-net に含まれる部品の他，JDK 1.2.2 および JDK 1.4.2 に含まれる部品が示されている．ここで，開発者がより新しいバージョンである JDK 1.4.2 を要求する場合を考える．開発者は，出自による絞り込み機能を用いて JDK 1.2.2 を除外することで，必要な利用関係のみを含むリストを得ることができる．JDK 1.2.2 を除外した結果の画面を図 8 に示す．同じ名前をもつ部品がそれぞれ 1 個に絞られていることが分かる．従来手法では，部品を個別に選択する必要があるために開発者の負担となると考えられる．

4.4 考　　察

本実験のデータセットに対しては，解析は十分に現実的な時間で完了した．しかし，表 2 から，利用関係解析の時間およびインディケーション数は，データセットのサイズに対して指数的に大きくなる可能性も示唆される．これは，名前の重複する部品が増えるに従い，1 個の参照に対するインディケーションおよびその属性の個数が増えることが原因であると考えられる．ソフトウェアリポジトリから複数の版の部品を収集する場合には，より大きな集合に対する解析となるため現実的な時間で終了しない，もしくは解析結果のデータベースが巨大になる可能性がある．

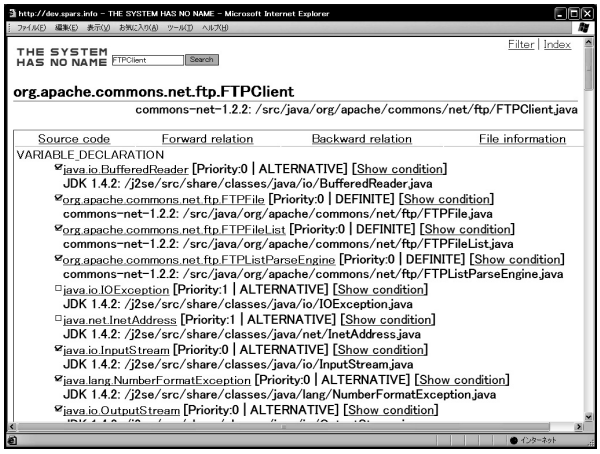

図 8　スクリーンショット: 利用関係 (2): 絞り込み結果
Fig. 8　Screenshot: use-relation (2): Filtering result

この問題への対処としては，全てのインディケーションを解析せず，例えば確信度が DEF であるものや優先度がある程度高いもののみに絞ることが考えられる．解析から除かれたインディケーションは，開発者が利用関係を要求した時に解析する．この方法により，開発者に対する利便性を保ったまま，解析時間を抑えることが出来ると考えられる．

また，異なる問題として，偶然に名前が一致した場合や元々は同じ部品であるが変更が大きい場合など，個別の参照の解析結果としては正しいが部品間の利用関係としては正しくない結果が含まれることがある．例えば，ある変数宣言の参照 f_v が e_{c1} もしくは e_{c2} を指し示し，関連するメソッド呼び出しの参照 f_m が，e_{c1} のメソッドである e_{m1} のみ指し示す時，e_{c2} に対する利用関係は誤りであると考えられる．ただし，多くのメソッド呼び出しのうち 1 個だけ欠けている場合などは，サンプルコードとして閲覧する場合など，実用上は問題無いことがある．そのため，同じ名前の部品に対する利用関係の候補が複数存在する場合には，開発者が適切に候補を選択できるように，それぞれ参照のインディケーションにどの程度含まれているか示すことが望ましいと考えられる．

5. 関連研究

5.1 再利用と利用関係

部品の再利用を行う際，その利用関係は重要な情報である．特に，オブジェクト指向言語では複数の部品の協調動作によりある機能を提供することが多く，部品の再利用が困難になることがある．例えば，文献 [8] では Factory パターンと呼ばれる設計をもつ部品群は理解が困難であることを実験により示している．文献 [2] では，ある部品を利用する為には，その部品が依存する部品を取得もしくは依存しないように部品を修正する必要があり，その労力が再利用を妨げていると指摘した上で，部品を再利用する際に必要となる依存関係を効率的に解決する手法を提案している．提案されている手法は，まず何らかの検索システムを用いて再利用可能な部品を含むソフトウェアを取得していることを前提としている．

5.2 ソフトウェア部品検索システム

Koders [6] および Google code search [9], Krugle [10] はインターネット上のソフトウェアリポジトリを用いて部品データベースを構築しているソフトウェア部品検索システムである．いずれも複数のプログラミング言語に対応している．Google code search はソフトウェアリポジトリの他にも WWW 上のサンプルコードやアーカイブからソースコードを収集している．また，正規表現を用いた柔軟な検索を提供している．これらのうち Koders のみが識別子の検索の形式で利用関係を利用できるが，単語による検索に基づいているため，提示される結果には多くの無関係な候補が含まれる．

SPARS-J [3]〜[5] は利用関係の解析機能をもつ部品検索システムである．いずれも利用関係を用いて部品の順位付けを行うことで，開発者は再利用性の高い部品を検索出来る．しかし，利用関係を解析する時に名前の重複は考慮されていない．同じ名前の部品が複数存在する場合に，いずれかのみを用いて利用関係を構築している．そのため，開発者が同じ名前の異なる部品に対する利用関係を必要とする時には，異なる手段を用いて特定する必要がある．

5.3 クロスリファレンサ

Ctags [11] は与えられたソースファイル集合に含まれる識別子の関係を解析するシステムであり，開発環境や検索システムに組み込んで利用される．LXR [12] および OpenGrok [13] は Ctags と全文検索システムを統合したシステムであり，オープンソースソフトウェアのソースコードの閲覧システムとして用いられることが多い．しかし，Ctags の解析は識別子に基づいているため，多数の候補の中から正しいものを選択する必要がある．また，特定のバージョンのソースコードのみが対象であることが多い．

5.4 ソフトウェアリポジトリ閲覧システム

ViewVC [14] は版管理システムのリポジトリを閲覧するためのシステムである．指定されたリポジトリの全ての版のソースコードおよびコミット時のログを閲覧することができ，一般には保守や理解の支援に用いられる．CREBASS [15] はクロスリファレンサ機能付きのリポジトリ閲覧システムである．時系列を考慮し，識別子間の参照関係を解析している．参照関係は時系列上で同時に存在するもの同士で構築される．

6. まとめと今後の課題

本稿では，名前の重複する部品集合における部品間の利用関係を解析する手法を提案した．提案手法では，参照の指し示す先となるエンティティの候補それぞれを属性とともに保存することにより，開発者による利用関係の適切な絞り込みを可能にする．また，提案手法を実装し，ソフトウェアリポジトリから部品を収集し，検索をおこなうシステムを提案した．さらに，試作したシステムを用いて適用実験を行い，既存手法の問題点が解決されることを確認した．

今後の課題は，まず，利用関係解析手法におけるスケーラビリティの向上が挙げられる．部品集合が大きくなると処理時間が極端に増大するため，より大きな部品集合を対象とするためには，利用関係の一部を検索システムの利用者の要求に応じて解析するなどの方法を用いることで解析量を減らすことが必要である．また，参照が指し示す部品の候補に実際には利用出来ない部品が含まれる問題へ対応することが望ましいと考えられる．最後に，提案した検索システムの実装を進め，より実際の再利用のシナリオに近い適用実験を行うことを予定している．

謝辞 本研究の一部は文部科学省「e-Society 基盤ソフトウェアの総合開発」の委託に基づいて行われた．また，本研究の一部は日本学術振興会科学研究費補助金萌芽研究（課題番号:18650006）の助成を得た．

文　献

[1] I. Jacobson, M. Griss, and P. Jonsson, Software Reuse, , 1997.
[2] R. Holmes, and R.J. Walker, “Supporting the investigation and planning of pragmatic reuse tasks,” Proc. 29th International Conference on Software Engineering (ICSE’07), pp.447-456, May 2007.
[3] K. Inoue, R. Yokomori, T. Yamamoto, M. Matsushita, and S. Kusumoto, “Ranking significance of software components based on use relations,” IEEE Trans. Software Engineering, vol.31, no.3, pp.213-225, March 2005.
[4] 横森励士，梅森文彰，西秀雄，山本哲男，松下誠，楠本真二，井上克郎，“Java ソフトウェア部品検索システム SPARS-J,” 電子情報通信学会論文誌 D-1，vol.J87-D-I，no.12，pp.1060-1068，2004.
[5] “SPARS-J,” http://demo.spars.info.
[6] “Koders,” http://www.koders.com.
[7] 谷口考治，“多言語対応メトリクス計測ツール masu の開発,” ウィンターワークショップ 2007・イン・那覇 論文集，pp.29–30，January 2007.
[8] B. Ellis, J. Stylos, and B. Myers, “The factory pattern in api design: A usability evaluation,” Proc. 29nd International Conference on Software Engineering (ICSE’07), pp.447–456, May 2007.
[9] “Google code search,” http://www.google.com/codesearch.
[10] “Krugle,” http://www.krugle.org.
[11] “Ctags,” http://ctags.sourceforge.net.
[12] “LXR,” http://lxr.linux.no.
[13] “OpenGrok,” http://opensolaris.org/os/project/opengrok.
[14] “ViewVC,” http://www.viewvc.org.
[15] 中山崇，松下誠，井上克郎，“関数の変更履歴と呼出し関係に基づいた開発履歴理解支援システム,” 信学技報，vol.104，no.47，pp.7–12，March 2004.