情報処理学会論文誌 Vol.54 No.2 857–869 (Feb. 2013)

ソースコードに出現するドメイン固有な動詞–目的語関係を収録した辞書作成手法

鹿島 悠[1,a)] 早瀬 康裕[2,b)] 眞鍋 雄貴[1,c)] 井上 克郎[1,d)]

受付日 2012年5月14日, 採録日 2012年11月2日

概要：プログラム理解において，識別子はプログラム要素をドメインの知識と対応させる重要な役割を果たしている．そのため，識別子に不適切な名前が付けられた場合，開発者はプログラムを理解するために多くの余計な時間を費やしてしまう．そこで，本研究ではオブジェクト指向プログラミング言語で記述されたソースコードから，良い命名の例として動詞–目的語関係を抽出し，メソッドの命名支援を目的とした辞書を作成する手法を提案する．実験では，いくつかのアプリケーションドメインを対象に動詞–目的語関係を収録した辞書を作成し，開発者が辞書を評価した．その結果，辞書に収録された関係が多くの場合適切であったことを確認した．

キーワード：プログラム理解，識別子，動詞–目的語関係，辞書，静的解析

Building Domain Specific Dictionaries of Verb-object Relation from Source Code

Yu Kashima[1,a)] Yasuhiro Hayase[2,b)] Yuki Manabe[1,c)] Katsuro Inoue[1,d)]

Received: May 14, 2012, Accepted: November 2, 2012

Abstract: An identifier is an important key in mapping program elements onto domain knowledge for the purpose of program comprehension. Therefore, if identifiers in a program have inappropriate names, developers can waste a lot of time trying to understand the program. This paper proposes a method for building a dictionary for supporting naming a method by extracting and gathering verb-object (V-O) relations, as good examples of naming, from source code written in an object-oriented programming language. For each of several application domains, dictionaries containing the V-O relations are built and evaluated by software developers. The evaluation results confirm that the relations in the dictionaries are adequate in many cases.

Keywords: program comprehension, identifier, verb-object relation, lexicon, static analysis

1. はじめに

ソフトウェア保守作業にかかる時間の半分以上を占めるといわれているプログラム理解 [1], [2] において，ソースコードに出現する識別子の果たす役割は大きい．これは，ソフトウェア開発者がプログラムを読み進める際に，識別子の意味からプログラム要素の役割を推測するためである [3], [4]．

そのため，ソフトウェア開発者はソースコード中の識別子に対しプログラム要素の役割を表す正確な名前を付けるのが望ましいとされている [5], [6]．もし，識別子にプログラム要素としての役割を表さない不適切な名前が付けられた場合，開発者がアプリケーションドメインの知識とプログラム要素を対応づけるのが困難になることが知られている．このことについて，Lawrie ら [7] は，識別子が頭文字

[1] 大阪大学大学院情報科学研究科
Graduate School of Information Science and Technology, Osaka University, Suita, Osaka 565–0871, Japan
[2] 筑波大学システム情報系
Faculty of Engineering, Information and Systems, University of Tsukuba, Tsukuba, Ibaraki 305–8573, Japan
a) y-kashima@ist.osaka-u.ac.jp
b) hayase@cs.tsukuba.ac.jp
c) y-manabe@ist.osaka-u.ac.jp
d) inoue@ist.osaka-u.ac.jp

だけの略語であったり意味のない連番である場合，識別子に略称を使わず正式な名前が使われている場合に比べて，開発者がプログラム理解に多くの時間を必要とすることを明らかにした．

しかし，識別子の役割を正確に表す名前を付けるには多くの知識や経験が必要であるため，すべての開発者が適切な名前を識別子に付けることはできない．このため，開発者は，プログラミング言語，開発組織，アプリケーションドメインといった異なるドメインにおける様々な単語やその組み合わせ方のルールを学ぶ必要がある．これらのルールは多くの場合明文化されていないため，実際の開発経験からルールを学ぶしかない．

そこで，我々の研究グループは識別子の命名支援を目的として，識別子の命名事例を収録したグラフ構造を持つ辞書の構築を行っている．以前の研究では，識別子に使われる名詞の上位下位関係（抽象–具体関係）を収録した辞書の構築手法を提案した [8]．この辞書は，クラス名や変数名といった，名詞で構成される識別子の命名に役立つと考えられる．

しかし，ソースコードに出現するのは名詞と名詞の上位下位関係だけではない．特に，メソッド名には，名詞だけではなく動詞も出現し，メソッドの振舞いを表現している．さらに，動詞の後ろに目的語句が続く場合があり，このとき，目的語句は振舞いの対象を表現している．上で述べた名詞の上位下位関係の辞書には，動詞は収録されておらず，また，動詞と目的語の関係についても収録されていない．そのため，メソッド名の命名にはあまり有効ではないと考えられる．

そこで本研究では，ソースコードから動詞–目的語関係を抽出し，抽出した関係を収録した辞書を構築する手法を提案する．本手法は，アプリケーションドメインごとに分類したソースファイル集合を入力とし，既存の自然言語処理技術と我々が開発したパターンマッチングシステムを用いて，動詞–目的語関係を抽出する．動詞–目的語関係はメソッドに関連する識別子から抽出し，具体的には，動詞をメソッド名から抽出し，目的語をメソッド名，仮引数名，メソッドが定義されているクラスの名前から抽出する．そして，抽出した動詞–目的語関係のうち，共通したドメインで頻出した関係をそれぞれのドメインの辞書に収録し，出力とする．提案手法により作成された辞書の活用法としては，プログラミング初心者や，開発しようとしているドメインに関する知識がない開発者に，辞書に収録された関係を提示することにより，メソッドの命名の支援を行うことを考えている [9]．

評価実験では，4 つの異なるドメインを扱う Java のソースファイル集合に対して提案手法を適用して辞書を生成し，その辞書を，Java プログラムの開発経験と辞書が対象とするドメインに関する知識を持つ被験者が評価した．その結果，辞書に含まれる関係のほとんどは，ドメイン固有の関係か，Java ソースコードで一般的な関係であることを確認した．

以降，2 章で一般的なオブジェクト指向プログラムにおける命名規約とメソッド名に含まれる動詞–目的語関係について説明する．3 章では動詞–目的語関係をソースコードから抽出するアルゴリズムについて述べる．4 章では評価実験とその結果を示し，5 章では，関連研究について議論し，6 章ではまとめと今後の課題を述べる．

2. オブジェクト指向プログラムにおける動詞–目的語関係

本章では，オブジェクト指向プログラムのメソッドにおける動詞–目的語関係について述べる．まず，オブジェクト指向プログラム，特に Java の識別子の命名規約を示す．そして，メソッドの宣言と関連する識別子内の単語間に現れる動詞–目的語関係について説明する．

2.1 識別子の命名規約

識別子にはその役割を示す明白な名前を付けるのが望ましいとされている [5], [6]．しかし，複雑な概念を扱うプログラムでは，識別子の役割もまた複雑になり，1 つの単語では識別子の役割が表せない場合がある．そのような場合，識別子名に複数の単語を用いることで複雑な役割を表現する．

多くのプログラミング言語で識別子に空白文字を含めることは許されていないため，識別子名は，空白文字の代わりにキャメルケースやスネークケースと呼ばれる方法を用いて，複数の単語をつなぎ合わせて表現される．キャメルケースでは，各単語の先頭文字を大文字にした後，すべての単語を空白文字なしで結合する（例：CamelCase）．スネークケースでは，単語を空白文字の代わりにアンダースコアで連結する（例：snake_case）．Java ソースコード中では，キャメルケースが推奨されている [10]．

また，オブジェクト指向プログラムのメソッドでは，メソッド名を動詞または動詞句から始めることが推奨されている [10], [11]．実際，メソッド名の先頭は動詞で後ろに名詞または形容詞が続くことが多いことが，Høst ら [12] の研究により明らかになっている．

一方で，メソッド名の先頭が名詞，形容詞，名詞句，または形容詞句であり，後ろに過去形の動詞が続くという場合もある．この例としては，JavaAPI [13] 中の java.awt.event.ActionListener というクラスの，actionPerformed(ActionEvent) というメソッドがあげられる．

一方で，少数ではあるが，メソッド名が動詞を含まない場合も存在する．たとえば，java.lang.Object のメソッド toString() や，java.lang.Class のメソッド newInstance() と

いったメソッド名には動詞が含まれていない．このようなメソッド名は，2 通りに解釈することができる．1 つ目の解釈は，動詞が省略されているという考え方であり，例の場合，convert や create といった動詞が省略されていると考えられる．2 つ目の解釈は，一部の単語が実質的に動詞の役割を果たしているという考え方である．例の場合，メソッド名中の to や new が動詞と見なすことができる．

2.2 メソッド中の動詞と目的語

オブジェクト指向プログラムでは，メソッドがオブジェクトに対して何らかの操作を行うという処理が頻出するが，操作と操作の対象であるオブジェクトの関係は，自然言語に出現する動詞と目的語の関係に類似している．このとき目的語に相当する名前が，メソッド名中の動詞の後や，メソッドを所有するクラスの名前，引数に出現するという特徴がある．この特徴に着目し，Fry ら [14] は，動詞と直接目的語の組をメソッドのシグネチャから抽出する手法を提案している．

上で述べたソースコードに現れる動詞と目的語の組には，自然言語で記述された文章には現れない組が含まれている場合がある．たとえば，java.net.Socket クラスには bind(SocketAddress) というメソッドがあり，このメソッドは，「bind SocketAddress to Socket（ソケットアドレスをソケットに束縛する）」という意味を表している．しかし，プログラムのドキュメント以外の自然言語の文章では，socket という語が bind という動詞の目的語となることはほとんどない．

また，ソースコード中に出現する動詞–目的語関係にはドメイン固有の関係が多く含まれている．これは，プログラムが扱うドメインによって，ソースコード中で使われる単語が異なっていたり，同じ単語でもドメインによって異なる意味を表したりする場合があるからである．ドメイン固有の関係の例としては，データベースの分野に頻出する「fetch from cursor」という動詞–目的語関係があげられる．この関係は，データベースからカーソルが指す一組のデータを取得することを意味している．ドメインごとに異なる意味を表す例としては，cursor という名詞があげられる．この名詞は，データベースの分野ではデータに 1 つずつアクセスするためのポインタを表している．一方で，GUI アプリケーションの分野ではテキストエリアの編集中の箇所を表すことが多い．

3. 動詞–目的語関係辞書構築手法の提案

本章では，オブジェクト指向プログラムのソースコードから動詞–目的語関係を抽出し，抽出した関係を収録した辞書を作成する手法について述べる．提案手法の入力はオブジェクト指向プログラミング言語で記述された，ある共通したドメインを扱う複数のソフトウェアのソースファイル集合であり，出力は動詞（V），直接目的語（DO），間接目的語（IO）の 3 つ組を収録した辞書である．ただし，IO については空の場合もある．提案手法では 3 つ組をソースファイル集合から抽出し，フィルタリングを行い，入力されたソースファイル集合で共通して出現する 3 つ組を取り出し辞書に収録する．

提案手法の概観を図 1 に示す．提案手法は図中の 3 つのステップで構成される．

ステップ 1：メソッドプロパティの取得　入力されたソースファイル集合にあるすべてのメソッド宣言を取り出

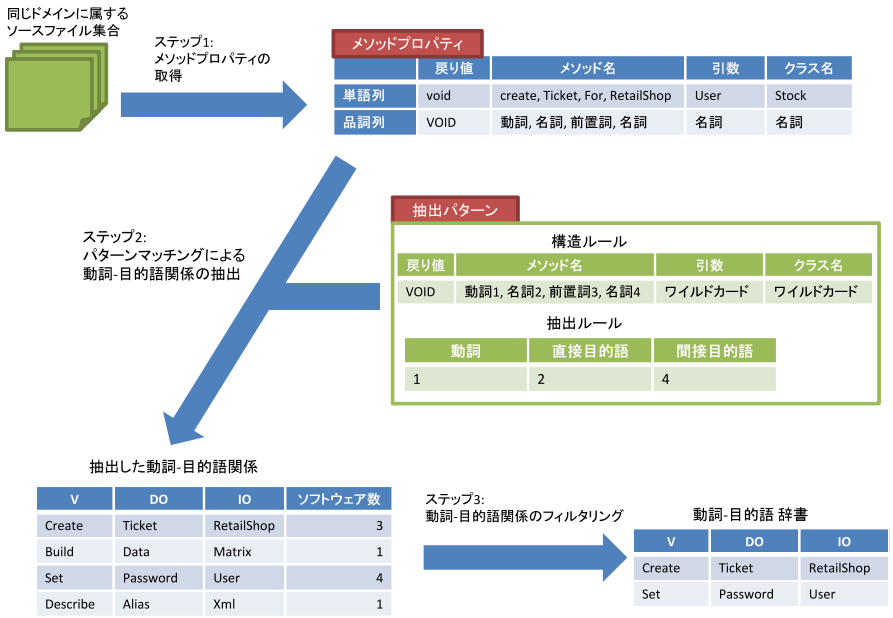

図 1　提案手法の概観

Fig. 1　Overview of our technique.

し，それぞれのメソッド宣言に関連する識別子（戻り値の型名，メソッド名，仮引数の名前と型名，メソッドが所属するクラス名）を取得する．そして，各メソッドに関連する識別子から単語を取り出し，それぞれの単語に品詞の情報を付与したもの（メソッドプロパティ）を出力する．

ステップ 2：パターンマッチングによる動詞–目的語関係の抽出　ステップ 1 の出力として得られたメソッドプロパティと事前に定義しておいた抽出パターンを用いてパターンマッチングを行い，<V，DO，IO> の 3 つ組を抽出する．

ステップ 3：動詞–目的語関係のフィルタリング　ステップ 2 の出力のうち，一定数以上のソフトウェアで出現する 3 つ組を抽出して辞書に収録する．

以降の節で各ステップについて詳しく述べる．

3.1 ステップ 1：メソッドプロパティの取得

本ステップでは，入力されたソースファイル中のすべてのメソッド宣言から関連する識別子を取得し，取得した識別子を解析して，メソッドプロパティを作成する．

メソッドプロパティとは，戻り値，引数，メソッド名，メソッドの定義されたクラスに対応する 4 要素の組である．また，各要素は単語列と品詞列の組である．品詞列は単語列中の対応する単語の品詞を示しており，品詞列中の n 番目の品詞は，単語列中の n 番目の単語の品詞を表す．図 1 中のメソッドプロパティは，表の 2 列目以降の列がそれぞれメソッドプロパティの 4 要素に対応しており，2 行目と 3 行目が各要素の単語列と品詞列に対応している．たとえば，メソッド名に対応する要素の単語列と品詞列はそれぞれ，(create, Ticket, For, RetailShop) と (動詞，名詞，前置詞，名詞) であり，create，Ticket，For，RetailShop の品詞はそれぞれ，動詞，名詞，前置詞，名詞である．なお，メソッドが 1 つ以上の引数を持つ場合には，メソッド 1 つに対し 2 つのメソッドプロパティが作成される．1 つは引数に対応する単語列に，引数の型名が入れられたメソッドプロパティが作成される．もう 1 つは，引数に対応する単語列に，仮引数名が入れられたメソッドプロパティが作成される．型が総称型で型パラメータが含まれている場合には，型パラメータの部分を無視する．たとえば，「List<Integer>」型の場合は，「List」のみを型名として扱う．**図 2** は 1 つのメソッドから 2 つのメソッドプロパティを作成している例である．具体的なメソッドプロパティの取得手順は後述するが，図中の引数の型名が User で仮引数名が customer であるメソッドから，型名を用いたメソッドプロパティと，仮引数名を用いたメソッドプロパティを取得している．2 つのメソッドプロパティは，3 番目の引数に対応する要素のみが異なっており，型名を用いたメソッドプロパティでは単語列が User であり，仮引数名を用いたメソッドプロパティでは単語列が customer となっている．

メソッド

戻り値	メソッド名	引数（型名：仮引数名）	クラス名
void	createTicketForRetailShop	User:customer	Stock

型名を用いたメソッドプロパティ

	戻り値	メソッド名	引数	クラス名
単語列	void	create, Ticket, For, RetailShop	User	Stock
品詞列	VOID	動詞, 名詞, 前置詞, 名詞	名詞	名詞

仮引数名を用いたメソッドプロパティ

	戻り値	メソッド名	引数	クラス名
単語列	void	create, Ticket, For, RetailShop	customer	Stock
品詞列	VOID	動詞, 名詞, 前置詞, 名詞	名詞	名詞

図 2　メソッドプロパティの取得例
Fig. 2　An example of creating method properties.

メソッドプロパティ取得の具体的な手順を述べる．まず，ソースファイルを解析し，すべてのメソッド宣言を抽出する．そしてそれぞれのメソッド宣言から，戻り値の型，メソッド名，仮引数の名前と型，メソッドが定義されているクラスの名前を抽出し，メソッドプロパティの各要素を作成する．

メソッドプロパティの 1 番目の要素である戻り値に対応した要素は，以下のように作成する．まず，単語列については，メソッドの戻り値の型名を 1 単語と見なし，この 1 単語のみを要素として持つ列を作成する．そして，品詞列については次のように作成する．戻り値が void 以外の場合には，戻り値の型名を名詞と見なして，1 個の「名詞」という要素を持つ列とする．戻り値が void の場合には，1 個の「VOID」特殊な品詞を要素として持つ列とする．

メソッドプロパティの 4 番目の要素であるメソッドが定義されたクラスに対応する要素は以下のように作成する．単語列は，クラス名を 1 単語として扱い，この 1 単語のみを要素として持つ列とする．ただし，ジェネリッククラスにおいて宣言されている総称型の型パラメータについては，型パラメータの記述を無視する．品詞列は，1 個の「名詞」という要素を持つ列とする．

メソッドプロパティの 3 番目の要素である引数に対応する要素は，以下のように作成する．引数が 0 個の場合は，単語列，品詞列ともに要素が空の列となる．引数が 1 個以上の場合は，仮引数名と型名を用いる場合で以下のように作成する．仮引数名を用いる場合，単語列は n 番目の仮引数の名前を要素として持つ列とする．型名を用いる場合，単語列は n 番目の引数の型名を要素として持つ列とする．ただし，型名に総称型の型パラメータが含まれる場合は，型名から型パラメータを削除する．品詞列についてはどちらを用いた場合も，n 個の「名詞」という要素を持つ列とする．

メソッドプロパティの 2 番目の要素であるメソッド名に対応する要素は以下の手順で作成する．初めに，メソッド名を単語列と見なし，キャメルケースとスネークケースに

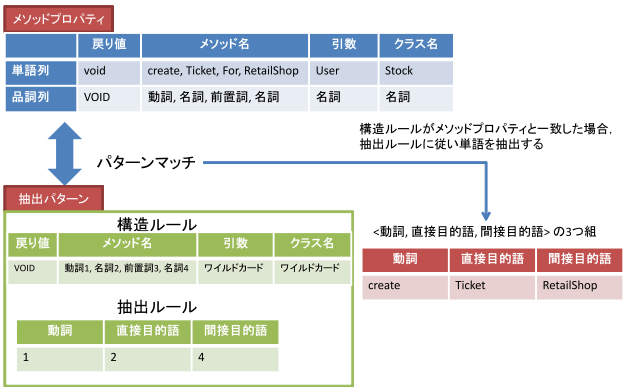

図 3 メソッドプロパティ，抽出パターン，パターンマッチ
Fig. 3 Method property, extraction pattern and pattern match.

従い，大文字の直前とアンダースコアで単語に区切って分割する．そして，各単語の品詞を自然言語処理を行うツールである OpenNLP [*1] を用いて特定する．ただし，メソッド名の先頭に to や new が出現する場合は，to や new を動詞と見なす．このように見なす理由は，2.1 節で述べた 2 通りの解釈のうち，動詞が省略されているという解釈を用いると，ソースコード中に出現しない動詞が辞書に含まれることになり，辞書を命名支援に利用する際，開発者に混乱を招くと考えたからである．最後に，動詞と前置詞以外の，連続して出現する単語を連結して名詞の 1 語とする．たとえば，図 2 のメソッド名の末尾の RetailShop という文字列は，キャメルケースに従って分割した後は Retail と Shop という 2 単語となっているが，この 2 単語を連結して RetailShop という名詞の 1 語とする．

3.2 ステップ 2：パターンマッチングによる動詞–目的語関係の抽出

本ステップでは，ステップ 1 で得られたメソッドプロパティと事前に人手で定義した抽出パターンを用いてパターンマッチングを行い，<V，DO，IO> の 3 つ組を取得する（**図 3** 参照）．

抽出パターンは，構造ルールと抽出ルールで構成される．構造ルールは 4 要素の組で表現され，各要素は，ワイルドカード，または，品詞と単語番号で構成される組の列のいずれかとなる．構造ルールの 4 要素はメソッドプロパティと同様に，戻り値の型，メソッド名，引数，クラス名にそれぞれ対応する．抽出ルールは単語番号の 3 つ組で，V，DO，IO にそれぞれ対応する．ただし，IO に対応する要素は空の場合がある．なお，以降表中の構造ルールのワイルドカードについては「*」と略記する．

パターンマッチングは以下の手順で行われる．まず，構造ルールとメソッドプロパティを照合し，次に示す 2 つの条件をすべて満たすか確認する．

- 構造ルールとメソッドプロパティの各 n 番目の要素について，構造ルールの要素がワイルドカードである，または，双方の要素の品詞の列が順序を含め完全に同一である．なお，メソッドが無名クラスに所属する場合には，メソッドプロパティのクラス名部は空になるため，構造ルールのクラス名部がワイルドカードのときのみマッチングに成功する．
- 構造ルールに同じ単語番号が複数箇所に出現している場合，メソッドプロパティ中の対応する単語が同じ単語である．

そして，条件が満たされている場合のみ，抽出ルールとメソッドプロパティを照合し，抽出ルールで V，DO，IO と指定された単語番号と対応しているメソッドプロパティ中の単語を抽出し，<V，DO，IO> の 3 つ組として出力する．ただし，抽出ルールの IO に対応する要素が空の場合は，IO は空になる．

パターンマッチングを行った結果，1 つのメソッドプロパティに対し，複数の抽出パターンの構造ルールが条件を満たすことがある．その場合は，それぞれの抽出パターンの抽出ルールを用いて，1 つのメソッドプロパティから複数の 3 つ組を抽出する．

最後に，3 つ組の抽出後，JWNL [*2] と WordNet [15] を用いて動詞の原型を検索し，3 つ組中の V に対応する単語を原型に変換する．

3.3 ステップ 3: 動詞–目的語関係のフィルタリング

本ステップはステップ 2 で得られた <V，DO，IO> の 3 つ組をフィルタリングし，ソースコードで頻出する動詞–目的語関係を収録した辞書を構築する．具体的には，入力のソースコード集合で事前に定めておいた閾値以上のソフトウェアで出現した 3 つ組のみを辞書に収録する．

4. 評価実験

提案手法によりドメイン固有の辞書が作成可能か確認するため評価実験を行った．評価実験では，まず，実験準備として抽出パターンを人手で作成しておき，作成したパターンを利用して 4 つのドメインの辞書を作成した．そして，被験者が辞書に収録された動詞–目的語関係を評価した．評価は，その関係がドメインで見られる組か，動詞と目的語に適切な語が選ばれているか，その関係をソフトウェア開発者に良い命名の例として見せてもよいか，という 3 つの観点に基づいて行った．以降，実験準備，評価対象の辞書，評価方法と実験結果，結果に対する議論を述べる．

4.1 実験準備

まず，辞書を作成するために**表 1** に示す 29 個の抽出パ

[*1] http://opennlp.sourceforge.net

[*2] http://sourceforge.net/projects/jwordnet/

表 1 作成した抽出パターン
Table 1 List of the extraction patterns.

	構造ルール				抽出ルール		
ID	戻り値	メソッド名	引数	クラス名	V	DO	IO
1	*	動詞 1 名詞 2 前置詞 3 名詞 4	*	*	動詞 1	名詞 2	名詞 4
2	*	動詞 1 前置詞 2 名詞 3	*	名詞 4	動詞 1	名詞 4	名詞 3
3	*	動詞 1 名詞 2	*	名詞 3	動詞 1	名詞 2	名詞 3
4	*	動詞 1 前置詞 2 名詞 3	名詞 4	*	動詞 1	名詞 4	名詞 3
5	*	動詞 1 名詞 2 前置詞 3	名詞 4	*	動詞 1	名詞 2	名詞 4
6	void	動詞 1	(空)	名詞 2	動詞 1	名詞 2	(空)
7	void	動詞 1 前置詞 2 名詞 3	(空)	名詞 4	動詞 1	名詞 4	名詞 3
8	void	動詞 1 名詞 2	名詞 2	名詞 3	動詞 1	名詞 2	名詞 3
9	void	動詞 1 名詞 2 前置詞 3 名詞 4	*	名詞 5	動詞 1	名詞 2	名詞 5
10	void	動詞 1	名詞 2	名詞 3	動詞 1	名詞 2	名詞 3
11	void	動詞 1 名詞 2	名詞 3	名詞 4	動詞 1	名詞 3	名詞 4
12	void	動詞 1 名詞 2	名詞 3	名詞 2	動詞 1	名詞 2	名詞 3
13	void	動詞 1 名詞 2	(空)	名詞 2	動詞 1	名詞 2	(空)
14	void	名詞 1 動詞 2	名詞 3	名詞 4	動詞 2	名詞 1	名詞 3
15	void	名詞 1 動詞 2	名詞 1	名詞 3	動詞 2	名詞 1	名詞 3
16	名詞 1	動詞 2 名詞 1	名詞 3	名詞 1	動詞 2	名詞 1	名詞 3
17	名詞 1	動詞 2 名詞 1 前置詞 3 名詞 4	名詞 4	名詞 5	動詞 2	名詞 1	名詞 4
18	名詞 1	動詞 2 名詞 3 前置詞 4 名詞 5	(空)	名詞 6	動詞 2	名詞 3	名詞 5
19	名詞 1	動詞 2 前置詞 3 名詞 4	名詞 5	名詞 6	動詞 2	名詞 6	名詞 4
20	名詞 1	動詞 2 名詞 1	(空)	名詞 3	動詞 2	名詞 1	名詞 3
21	名詞 1	動詞 2	名詞 3	名詞 4	動詞 2	名詞 4	名詞 3
22	名詞 1	動詞 2 前置詞 3	名詞 4	名詞 5	動詞 2	名詞 5	名詞 4
23	名詞 1	動詞 2 前置詞 3 名詞 4	*	*	動詞 2	名詞 1	名詞 4
24	名詞 1	動詞 2 前置詞 3 名詞 4	(空)	名詞 1	動詞 2	名詞 1	名詞 4
25	名詞 1	動詞 2 前置詞 3	名詞 4	名詞 4	動詞 2	名詞 4	名詞 4
26	名詞 1	動詞 2 名詞 3	(空)	名詞 1	動詞 2	名詞 3	名詞 1
27	名詞 1	動詞 2	(空)	名詞 3	動詞 2	名詞 3	(空)
28	名詞 1	動詞 2 名詞 3	(空)	名詞 3	動詞 2	名詞 3	名詞 1
29	名詞 1	動詞 2 名詞 1	(空)	名詞 1	動詞 2	名詞 1	(空)

ターンを我々が作成した．表中の 1 列目は論文中の各抽出パターンを識別するための ID である．抽出パターンの作成手順は以下のとおりである．まず，我々の経験から <V, DO, IO> の 3 つ組が明白なメソッドシグネチャを想定し，getter を想定した抽出パターン 20 や setter を想定した抽出パターン 8 など基本となる抽出パターンをいくつか定義した．そして，あるソフトウェア集合に対して定義済みの抽出パターンでは，3 つ組の抽出が行えないメソッドを列挙し，それらのメソッドから 3 つ組を抽出できるような抽出パターンを定義するということを繰り返した．

抽出パターンの作成後，**表 2** に示す 38 個のオープンソースソフトウェアを入力として，辞書を作成した．これらのソフトウェアは，ウェブアプリケーション（Web），XML 処理（XML），データベース（DB），GUI の 4 つのドメインに我々が事前に分類した．そして，それぞれのドメインに属するソフトウェアを入力として，ソースコードを解析し入力と対応するドメインの辞書を作成した．

4.2 評価対象の辞書

表 3 に提案手法のステップ 3 で行うフィルタリング前の解析結果を示す．なお，1 つのメソッドが複数の抽出パターンとマッチして，複数の 3 つ組を生成することもあるため，3 つ組の数は抽出パターンにマッチしたメソッドの数よりも多くなっている．

そして，**表 4** に，実験対象とした各ドメインにおける，3 つ組の出現頻度の分布を示す．この表より，2 つ以上のソフトウェアで出現する 3 つ組の数は，1 つのソフトウェアでしか出現しない 3 つ組の数に比べ，大きく減少していることが分かる．

本実験ではフィルタリングの閾値を 3 以上にすると辞書に収録される 3 つ組の数が無作為抽出を行うのに不十分となると考え，フィルタリングの閾値を 2 と定めた．つまり，2 つ以上のソフトウェアで出現した 3 つ組を収録した辞書を作成した．

また，**表 5** に各抽出パターンにより 3 つ組の抽出に成功したメソッドの数を示す．表中の括弧内の数値は各ソフト

表 2 抽出対象のソフトウェア
Table 2 Targets of extraction.

Web Applications	
BBS-CS 8.0.3	JForum 2.1.8
JGossip 1.1.0.005	mvnForum 1.2.1
Yazd Discussion Forum Software 3.0	Order Portal 1.2.4
Arianne RPG 0.80	JBoss Wiki Beta2
JSP Wiki 2.8.3	SnipSnap 1.0b3
XML	
Castor 1.3	DOM4J 1.6.1
JDOM 1.1.1	Piccolo 1.04
Saxon-HE 9.2.0.5	Xalan-J 2.7.1
Xbeans 2.0.0	Xerces-J 2.9.0
XOM 1.2.4	XPP3 1.1.4
xstream-1.3.1	
Databases	
Axion 1.0 Milestone 2	Apache Derby 10.5.3
H2 1.2.128	HSQLDB 1.8.1.1
Berkeley DB Java Edition 4.0.92	Mckoi 1.0.3
MyOODB 4.0.0	NeoDatis 1.9.22.674
OZONE 1.1	tinySQL 2.26
GUIs	
ArgoUML 0.28.1	BlueJ 2.5.3
Eclipse Classic 3.5.1	jEdit 4.3.1
NetBeans 6.8	vuze 4.3.1.2
LimeWire 5.4	

表 3 メソッド数と抽出した 3 つ組の数
Table 3 Number of methods and extracted tuples.

	メソッド数	1 つ以上のパターンとマッチしたメソッドの数	マッチしたメソッドの割合	3 つ組の数
Web	74,707	67,276	90%	67,429
XML	55,812	46,885	84%	49,926
DB	74,127	60,326	81%	63,087
GUI	298,696	247,918	83%	273,202

表 4 3 つ組の出現頻度の分布
Table 4 Frequently distribution of tuples.

	3 つ組が出現したソフトウェアの数					
	1	2	3	4	5	6
Web	67,147	258	18	4	2	0
XML	49,379	465	63	13	5	1
DB	62,415	609	28	1	32	2
GUI	272,795	339	38	23	5	2

ウェア集合の全メソッド数に対して抽出に成功したメソッドの数の比を表している．表より抽出パターン 3 が半分以上のメソッドから 3 つ組を抽出していることが分かる．また，抽出パターン 11 も抽出パターン 3 に次いで多くのメソッドから 3 つ組を抽出している．

4.3 評価方法

評価対象の辞書をソフトウェア工学の研究室の学生 6 名が 3 つの観点に基づいて評価した．被験者は全員 Java でのソフトウェア開発の経験を有している．加えて，それぞれの被験者が評価した辞書は，被験者が知識を持つドメインのものとした．

各ドメインの辞書について，収録されている <V，DO，IO> の 3 つ組を以下の観点で評価した．

初めに，収録された 3 つ組が属するドメインを確認する観点を用意した．作成した辞書は，対象ドメイン固有の 3 つ組を含むだけではなく，より広いドメイン（対象ドメインを包含する上位ドメイン）に所属する 3 つ組が含まれる可能性がある．そこで，評価対象の 3 つ組が，対象ドメインに所属するものなのかどうかを確認するだけではなく，より広いドメインとして抽出対象のソフトウェアすべてに含まれる Java プログラム全体に所属するものであるかどうかを評価することにした．

観点 1 3 つ組の動詞–目的語関係は実際にそのドメインで使われる関係であるか．あるいは Java プログラムで共通して使われる関係であるか．

次に，収録された 3 つ組が不適切な単語や不適切な組合せである可能性を考慮し，以下の観点を用意した．

観点 2 動詞，直接目的語，間接目的語は適切か．

最後に，辞書の活用例として考えている命名支援に，収録された 3 つ組が役立つかどうかを確かめるために以下の観点を用意した．

観点 3 3 つ組が識別子の適切な命名に役立つか．

以上の 3 つの観点を基に被験者への質問を用意した．まず，観点 1 に対応する質問として以下の 3 つを設定した．

Q1 この <V，DO，IO> の 3 つ組は，辞書が対象としているドメインでよく見られるか？

Q2 この <V，DO，IO> の 3 つ組は，Java プログラム一般でよく見られるか？

Q3 この <V，DO，IO> の 3 つ組は，他のドメインでよく見られるか？ そうならば，ドメイン名を答えよ．

次に，観点 2 に対応する以下の質問を設定した．

Q4 V，DO，IO はそれぞれ正しく抽出されているか？ そう思わないならば，不正確だと思う語を明示せよ．

最後に，観点 3 に対応する質問として以下の 3 つを設定した．

Q5 この <V，DO，IO> の 3 つ組を，辞書が対象とするドメインを扱うプログラムの開発者に，良い命名の例として見せてもよいと思うか？

Q6 この <V，DO，IO> の 3 つ組を，一般的な Java プログラムの開発者に，良い命名の例として見せてもよいと思うか？

Q7 この <V，DO，IO> の 3 つ組を，他のドメインを扱うプログラムの開発者に，良い命名の例として見せても

表 5 各抽出パターンにより 3 つ組が抽出されたメソッドの数
Table 5 The number of methods extracted tuples by extraction pattern.

ID	DB	GUI	WEB	XML
1	2,782 (3.75%)	7,535 (2.52%)	4,188 (5.61%)	1,714 (3.07%)
2	1,612 (2.17%)	4,300 (1.44%)	1,165 (1.56%)	757 (1.36%)
3	49,951 (67.39%)	201,859 (67.58%)	59,015 (79.00%)	39,540 (70.84%)
4	492 (0.66%)	1,826 (0.61%)	482 (0.65%)	327 (0.59%)
5	114 (0.15%)	875 (0.29%)	126 (0.17%)	124 (0.22%)
6	2,048 (2.76%)	9,217 (3.09%)	798 (1.07%)	1,091 (1.95%)
7	309 (0.42%)	561 (0.19%)	346 (0.46%)	92 (0.16%)
8	2,072 (2.80%)	11,752 (3.93%)	4,994 (6.68%)	3,679 (6.59%)
9	1,423 (1.92%)	2,882 (0.96%)	2,891 (3.87%)	962 (1.72%)
10	1,218 (1.64%)	6,372 (2.13%)	607 (0.81%)	1,089 (1.95%)
11	5,865 (7.91%)	34,064 (11.40%)	4,803 (6.43%)	5,384 (9.65%)
12	18 (0.02%)	47 (0.02%)	8 (0.01%)	11 (0.02%)
13	34 (0.05%)	38 (0.01%)	9 (0.01%)	8 (0.01%)
14	474 (0.64%)	10,826 (3.62%)	347 (0.46%)	230 (0.41%)
15	17 (0.02%)	1,092 (0.37%)	10 (0.01%)	3 (0.01%)
16	16 (0.02%)	35 (0.01%)	7 (0.01%)	28 (0.05%)
17	27 (0.04%)	101 (0.03%)	98 (0.13%)	15 (0.03%)
18	587 (0.79%)	1,651 (0.55%)	363 (0.49%)	166 (0.30%)
19	152 (0.21%)	719 (0.24%)	158 (0.21%)	225 (0.40%)
20	2,829 (3.82%)	7,310 (2.45%)	1,278 (1.71%)	2,461 (4.41%)
21	1,239 (1.67%)	6,358 (2.13%)	523 (0.70%)	1,363 (2.44%)
22	85 (0.11%)	408 (0.14%)	17 (0.02%)	92 (0.16%)
23	604 (0.81%)	2,233 (0.75%)	549 (0.73%)	493 (0.88%)
24	2 (0.00%)	7 (0.00%)	0 (0.00%)	1 (0.00%)
25	23 (0.03%)	43 (0.01%)	4 (0.01%)	14 (0.03%)
26	139 (0.19%)	849 (0.28%)	67 (0.09%)	211 (0.38%)
27	774 (1.04%)	1,886 (0.63%)	367 (0.49%)	814 (1.46%)
28	22 (0.03%)	81 (0.03%)	14 (0.02%)	28 (0.05%)
29	7 (0.01%)	22 (0.01%)	0 (0.00%)	10 (0.02%)

よいと思うか？ そうならば，そのドメインを答えよ．

回答の方法であるが，Q1，Q2，Q5，Q6 について，被験者は，(A) 強く同意する，(B) 同意する，(C) 同意しない，(D) 強く同意しない，の 4 段階評価あるいは，(Z) 分からない，で回答した．

各辞書は 2 名の被験者により評価され，各被験者は 2 つの辞書を評価した．被験者は 1 つの辞書に対し 30 個の 3 つ組を評価する．そのうち 15 個は 3 つ以上のソフトウェアで出現した 3 つ組で，もう 15 個は 2 つのソフトウェアで出現した 3 つ組である．各 3 つ組は辞書から条件に合致するものを無作為かつ排他的に抽出した．よって，1 つの 3 つ組を複数の被験者が評価することはないようにした．しかし，Web Application の辞書については，3 つ以上のソフトウェアで出現した 3 つ組は 24 個しかなかったため，無作為に選んだ 6 個の 3 つ組を 2 名の被験者が回答した．

4.4 実験結果

まず，観点 1 に対応する質問の回答結果について述べる．**表 6** に Q1 の結果を，**表 7** に Q2 の結果を示す．Q1 の回答における (A) 強く同意すると (B) 同意するの合計は 62%から 75%であり，また Q2 については 38%から 76%である．以上の結果は，辞書に含まれる関係の多くがドメイン固有の関係であることを示している．しかし，ドメイン

表 6 Q1 への回答
Table 6 Response to Q1.

	(A)	(B)	(C)	(D)	(Z)
Web	21	35	7	3	24
XML	44	18	5	7	16
DB	32	36	4	9	9
GUI	42	26	9	6	7

表 7 Q2 への回答
Table 7 Response to Q2.

	(A)	(B)	(C)	(D)	(Z)
Web	16	29	10	30	5
XML	15	11	17	42	5
DB	22	13	32	17	6
GUI	32	37	9	6	6

表 8 Q3 の回答（括弧内の数字は同じ回答の数を表す）
Table 8 Response to Q3 (Numbers in parentheses mean the number of the same answers).

Web	データベース (16)，入出力 (6)， Java プログラム一般 (2)
XML	データ解析 (2)，GUI (1)， パーサー (1)，リソース管理 (1)， 木構造 (1)，グラフ処理 (1)
DB	GUI (5)，Web アプリケーション (1)， 文字列処理 (1)
GUI	データベース (1)，ネットワーク (1)， プログラムテストケース (1)，アーカイブ (1)， Java プログラム一般 (4)

表 9 Q3 においてその他のドメインで見られると回答された組
Table 9 Tuples being popular in other domains at responses of Q3.

	動詞	直接目的語	間接目的語	見られると答えられたドメイン
Web	Set	Password	User	データベース処理
XML	Perform	Action	ActionEvent	GUI
DB	Release	Mouse	MouseEvent	GUI
GUI	Open	Connection	Handler	データベース，ネットワーク関係

表 10 Q4 への回答
Table 10 Response to Q4.

	動詞	直接目的語	間接目的語	2 つ以上
Web	3	1	3	6
XML	5	7	1	12
DB	1	6	5	11
GUI	8	1	0	9

とは無関係な関係も含まれていたことも示している．

表 8 に Q3 の回答を示している．すべての辞書がその他のドメインに属する動詞–目的語関係を含んでいた．そのため，辞書に含まれるその他のドメインを分離することで，辞書を改善できると考えられる．Q3 の回答の具体例については，**表 9** に示す．表 9 のとおり，DB の辞書に対して <Release，Mouse，MouseEvent> のような GUI のドメインに属する組が混入するなどの例が見られた．

次に，観点 2 に対応する質問の回答結果を示す．**表 10** に Q4 の回答を示す．不正確な V，DO，IO の割合は 6%から 13%であった．このことは，作成した 29 個の抽出パターンには改善の余地があることを示している．Q4 で不適切と判断された組の具体例について，**表 11** に示す．表 11 より，DB や XML の例のように Dtd や Evt のような省略語を用いた組が不適切と判断されていることが分かる．また，GUI の例のようにメソッド先頭の To を動詞として判定した我々の判断は被験者に不適切と判断された場合もあった．

表 11 Q4 において不適切だと回答された組
Table 11 Tuples being incorrect at responses of Q4.

	動詞	直接目的語	間接目的語	不適切だと判断された箇所
Web	Page	Exists	WikiEngine	動詞，間接目的語
XML	Start	Dtd	XmlWriter	直接目的語
DB	Perform	Action	Evt	間接目的語
GUI	To	String	Mode	動詞

表 12 Q5 への回答
Table 12 Response to Q5.

	(A)	(B)	(C)	(D)	(Z)
Web	19	32	11	4	24
XML	33	15	10	16	16
DB	35	29	10	10	6
GUI	28	30	13	11	8

表 13 Q6 への回答
Table 13 Response to Q6.

	(A)	(B)	(C)	(D)	(Z)
Web	13	19	23	30	5
XML	14	13	12	46	5
DB	31	13	24	16	6
GUI	29	26	15	13	7

表 14 Q7 の回答（括弧内の数字は同じ回答の数を示す）
Table 14 Response to Q7 (Numbers in the parenthese mean the number of the same answers).

XML	データ解析 (1)，GUI (1)，パーサー (1)， リソース管理 (1)，木構造 (1)，グラフ処理 (1)
DB	GUI (5)，ウェブアプリケーション (1)

表 15 対象ドメインにおいて有用であると判定された組
Table 15 Tuples evaluated useful at the target domain.

	動詞	直接目的語	間接目的語
Web	Destroy	Session	HttpSessionEvent
XML	Declare	Prefix	NamespaceSupport
DB	Add	Constraint	Table
GUI	Click	Mouse	MouseEvent

さらに，観点 3 に対応する質問の回答結果を示す．**表 12** に Q5 の結果を，**表 13** に Q6 の結果を示す．また Q1 と Q5 において，対象ドメインにおいて有用だと判定された組については，具体例を**表 15** に示す．Q5 の回答における (A) 強く同意すると (B) 同意するの合計は 53%から 71%であり，また Q6 については 30%から 61%である．Q5 と Q6 の結果は，辞書を改善するには，辞書の精度を上げるだけでなく，開発者に役立つ関係を選択し収録する必要があることを示している．一方，Q6 の結果は，辞書には Java プログラム一般で見られ，命名の役に立つ良い例が多く収録

表 16 抽出パターンと抽出された 3 つ組に対する評価（セル内の数値は Q1/Q2/Q5/Q6 での評価に対応）

Table 16 Extraction patterns and responses of tuples (numbers in each cell correspond to the responses of Q1/Q2/Q5/Q6).

ID	A	B	C	D	Z
1	7/3/9/4	17/3/13/3	1/15/5/12	2/10/3/12	7/3/4/3
2	1/0/1/0	0/2/0/1	1/0/1/1	0/0/0/0	1/1/1/1
3	78/49/65/52	66/45/58/35	11/36/20/36	15/62/27/69	34/12/34/12
4	5/3/3/3	0/1/1/1	0/0/1/0	0/1/0/1	0/0/0/0
5	2/1/1/1	2/3/1/2	0/0/2/1	1/3/1/3	2/0/2/0
6	8/4/7/6	5/6/6/4	2/4/2/4	0/2/0/2	1/0/1/0
7	0/0/0/0	0/0/0/0	0/0/0/0	0/0/0/0	0/0/0/0
8	14/10/12/10	12/9/11/7	2/6/5/8	2/7/2/7	3/1/3/1
9	0/0/0/0	0/0/0/0	0/0/0/0	0/0/0/0	0/0/0/0
10	1/0/0/0	1/0/2/0	0/1/0/0	0/3/0/4	4/2/4/2
11	9/8/9/7	7/4/6/4	3/2/3/3	1/8/2/8	2/0/2/0
12	0/0/0/0	0/0/0/0	0/0/0/0	0/0/0/0	0/0/0/0
13	0/0/0/0	0/0/0/0	0/0/0/0	0/0/0/0	0/0/0/0
14	30/13/22/12	16/25/20/18	3/9/4/16	3/7/5/7	4/2/5/3
15	0/0/0/0	0/0/0/0	0/0/0/0	0/0/0/0	0/0/0/0
16	0/1/0/1	1/0/1/0	0/0/0/0	0/0/0/0	0/0/0/0
17	1/0/1/1	1/0/1/0	0/2/0/0	0/0/0/1	0/0/0/0
18	4/3/5/3	10/0/9/0	1/11/4/10	2/6/2/7	5/2/2/2
19	0/0/0/0	0/0/0/0	0/0/0/0	0/0/0/0	0/0/0/0
20	7/10/6/8	9/7/3/3	2/1/4/1	3/5/8/11	2/0/2/0
21	3/3/3/3	1/3/1/3	2/0/2/0	0/0/0/0	2/2/2/2
22	0/0/0/0	0/0/0/0	0/0/0/0	0/0/0/0	0/0/0/0
23	1/3/1/2	3/2/1/3	1/1/3/1	3/3/3/3	2/1/2/1
24	0/0/0/0	0/0/0/0	0/0/0/0	0/0/0/0	0/0/0/0
25	0/0/0/0	0/0/0/0	0/0/0/0	0/0/0/0	0/0/0/0
26	0/0/0/0	1/0/1/0	0/1/0/0	0/0/0/1	0/0/0/0
27	1/2/1/1	0/1/0/2	2/0/2/0	0/1/0/1	2/1/2/1
28	0/0/0/0	0/0/0/0	0/0/0/0	0/0/0/0	0/0/0/0
29	0/0/0/0	0/0/0/0	0/0/0/0	0/0/0/0	0/0/0/0

されていることを示している．これらの組は，ドメインを対象とした辞書とは別の辞書に分けて収録するべきだと思われる．また，**表 14** に Q7 の結果を示す．Q3 の結果と同様，Q7 の結果からも，辞書には他のドメインの辞書に分離するべき組が収録されていたことを確認した．

さらに，各抽出パターンが有用な 3 つ組の抽出に貢献していたかを確認するため，抽出パターンとその抽出パターンにより抽出された 3 つ組に対する評価との関連を調査した．**表 16** は，各抽出パターンにより抽出された 3 つ組が，Q1，Q2，Q5，Q6 において受けた評価を示している．1 列目は抽出パターンの ID であり，2 列目以降の各セル内の数値は，1 行目に記述されている評価を受けた 3 つ組の数について，Q1/Q2/Q5/Q6 のそれぞれの回答における評価の数という形式で表している．表より，高い評価を受ける 3 つ組の抽出に最も貢献したのは抽出パターン 3 であり，これはメソッドの抽出に最も貢献したためであると考えられる．また抽出パターン 14 や抽出パターン 20 など，3 つ組を抽出できたメソッドが少なくても評価の高い 3 つ組の抽出に貢献していたものもあった．

4.4.1 結果に対する追加調査

実験結果を精査したところ，いくつかの 3 つ組は辞書が対象とするドメインに属するプログラム，または Java プログラム一般で見られるにもかかわらず，開発者にとって望ましいものではないと評価されていることが分かった．そこで，被検者に対して，望ましくないと判断した理由について追加の質問を行った．その結果，以下の回答が得られた．

- 3 つ組に省略語など意味が不明瞭な語が使われていた．
- 3 つ組が平均的な技能を持つ開発者にとって常識的なものだった．
- 3 つ組はドメイン全体で使われるものではなく，ある特定のライブラリに依存したプログラムにだけ出現するものだった．

以上の回答から，辞書には，省略語が使われた 3 つ組，

常識的な 3 つ組，特定のライブラリに依存した 3 つ組といった，開発者にとって提示しても有用とはいえない関係が収録されていたことが分かった．辞書をより有用なものにするには，これらの 3 つ組に対して何らかの対応が必要である．省略語が使われた 3 つ組については，辞書中に 3 つ組に出現した省略語についての説明を加えるという対応が考えられる．

常識的な 3 つ組については，辞書から取り除く必要はないと考えられる．なぜなら，平均的な開発者が皆知っている常識的な関係というのは，初心者にとっては有用な情報であると考えられるからである．

特定のライブラリに依存した 3 つ組については，辞書作成の際の入力となるソフトウェアを増やし，フィルタリングの閾値も大きくして，ライブラリに依存した 3 つ組が辞書に収録されないようにするという対応が考えられる．

4.5 議論

Q2 と Q3 の結果から別のドメインに属する動詞–目的語関係が辞書に含まれることが分かった．この理由は以下の 2 つが考えられる．

- フィルタリングに使用した閾値がノイズを取り除くのには小さすぎた．
- ソフトウェアは一般的に複数のドメインを扱うことがあり，異なるソフトウェアで意図せず同じドメインを扱っていた可能性がある．

1 つ目の問題を解決する単純な方法は，入力するソフトウェアの数を増やし，フィルタリングの閾値も大きくすることである．

2 つ目の問題を解決する方法としては，我々は以下に示す方法を考えている．まず，ドメインの数と入力するソフトウェアの数を増やし，ソフトウェアをドメインごとに非排他的に分類する．そして，3 つ組を抽出し辞書を作成したのち，各 3 つ組の抽出元のソフトウェアを特定する．もし，同じ 3 つ組が複数の辞書に出現した場合，抽出元のソフトウェアが所属するドメインを基に，最も多くのソフトウェアが共通して扱っているドメインを特定する．最終的に，そのドメインを 3 つ組を収録するのに最善の辞書と判断し，他の辞書からはその 3 つ組を除外する．

4.6 妥当性への脅威

評価対象となった 3 つ組は辞書から無作為に抽出したため，データは全標本から無作為抽出したものとなっている．しかし，辞書を作成する際の入力となったソフトウェアは無作為に収集したものではなく，ソフトウェアの数も不十分な可能性がある．そのため，本実験で作成した辞書は，提案手法で作成可能な辞書として，十分に一般的でない可能性がある．しかし，ソフトウェアは意図的に実験結果を制御するために集めたものではなく，あくまでドメインごとに分類するために意図的な収集を行ったものである．

被験者については，彼らは大学院生でありソフトウェア開発の専門家ではない．しかし，彼らはアルバイトや研究活動でのプログラミング経験があり，辞書のドメインについての十分な知識を有している．ただし，被験者の数は十分ではなかったため，辞書に対し被験者を無作為に割り当てることはできなかった．

5. 関連研究

本章は，ソースコード中の動詞–目的語関係についての関連研究を示し，本研究との違いについても述べる．

Fry ら [14] はオブジェクト指向プログラムのソースコード中のメソッドから動詞と直接目的語の組を抽出する方法を示した．Shepherd ら [16] はメソッドから抽出した動詞と直接目的語の組を用いて，Action-Oriented Identifier Graph と呼ばれるグラフを作成し，フィーチャーロケーションやアスペクトマイニングに利用する手法を提案した．Hill ら [17] は，動詞と直接目的語に加え，間接目的語もメソッドから抽出し，< 動詞，直接目的語，間接目的語 > の 3 つ組を利用して，ソースコードに対する検索クエリを開発者に提示する手法を提案している．我々の手法との相違点としては，彼らの手法が単一のソフトウェアを対象にしていることに対して，我々の手法は複数のソフトウェアを対象にしている点と，フィルタリングによりドメイン固有の関係を取り出している点があげられる．ただし，我々の手法のメソッドから動詞–目的語関係を抽出する部分についてはこれらの手法を参考にしている．

Høst ら [18] はメソッド名に含まれる動詞とメソッドの特徴を基に，メソッド名に用いられる動詞の辞書を作成した．この辞書は，メソッド名に用いられる代表的な動詞 40 個について，その動詞が使われるメソッドが持つ特徴と特徴を持つ頻度を収録した辞書となっている．たとえば，get という動詞については，「非常に多くの場合状態の読み込みを行い，引数を持つことは稀である」といった特徴と頻度が記述してある．我々の手法は，辞書に動詞だけでなく目的語が含まれている点が大きく異なっている．そのため，我々の作成した辞書からは，動詞だけでなく目的語まで含めてメソッド名を提案することができる．

Zhou ら [19] は，ソースコードに出現する is-a 関係と has-a 関係をソフトウェアのクラス図から抽出し，抽出した関係と，人手で作成されたドメインの is-a 関係と has-a 関係の知識とを統合し，プログラム理解や設計ミスの発見に役立てる手法を提案している．Zhou らの研究とは異なり，我々の研究の目的はドメインの知識を作成することである．加えて，本研究でドメインの知識として収録される関係は，is-a 関係や has-a 関係ではなく，動詞–目的語の関係である．Zhou らの手法で使用されるドメインの知識に，我々の手法で作成された辞書をあわせて用いることで，彼

らの手法では扱っていなかった関係もあわせて表示されるようになり，プログラムや設計ミスの発見により役立つことが期待される．

6. まとめと今後の課題

本研究はドメイン固有の動詞–目的語関係を収録した辞書を作成する手法を提案した．辞書に収録されているのは，<V，DO，IO> で構成される 3 つ組であり，それらはメソッドに関連する識別子から収集したものである．

評価実験では，29 個の抽出パターンを人手で作成し，4 つのドメインの辞書を作成し，6 名の被験者が辞書を評価した．その結果辞書に含まれる関係の多くが，辞書が対象とするドメインまたは Java プログラム一般で見られる関係であったことを確認した．さらに，開発者に良い命名の例として見せてもよい関係が多く収録されていることも確認した．

今後の課題としては，辞書の精度を上げること，大規模なソースコード集合を対象に，より大きな辞書を作成することがあげられる．また，命名支援のために，ソースコードのコンテキストにあわせて開発者に対し良い名前の例を提示するシステムの開発も必要である．良い命名を支援するためには，辞書をドメインの知識がない開発者に対して視覚化することが有用であると考えられる．

謝辞 本研究は，日本学術振興会科学研究費補助金基盤研究（A）（課題番号：21240002）と文部科学省科学研究費補助金若手研究（B）（課題番号：21700031）の助成を受けた．

参考文献

[1] Fjeldstad, R.K. and Hamlen, W.T.: Application Program maintenance study: Report to our respondents, *Proc. GUIDE 48* (1983).
[2] Corbi, T.A.: Program understanding: Challenge for the 1990's, *IBM Syst. J.*, Vol.28, No.2, pp.294–306 (1989).
[3] von Mayrhauser, A. and Vans, A.M.: Identification of Dynamic Comprehension Processes During Large Scale Maintenance, *IEEE Trans. Softw. Eng.*, Vol.22, No.6, pp.424–437 (1996).
[4] Pennington, N.: *Empirical studies of programmers: 2nd workshop*, Ablex Publishing Corp., pp.100–113 (1987).
[5] McConnell, S.: *Code Complete, Second Edition*, Microsoft Press, Redmond, WA, USA (2004).
[6] Martin, R.C.: *Clean Code: A Handbook of Agile Software Craftsmanship, 1st edition*, Prentice Hall PTR, Upper Saddle River, NJ, USA (2008).
[7] Lawrie, D., Morrell, C., Feild, H. and Binkley, D.: What's in a Name? A Study of Identifiers, *Proc. 14th IEEE International Conference on Program Comprehension*, pp.3–12 (2006).
[8] 早瀬康裕，市井 誠，井上克郎：ソフトウェア理解支援を目的とした辞書の作成法，*Proc. Winter Workshop 2008 in Dogo*, No.3, pp.33–34 (2008).
[9] 鬼塚勇弥，早瀬康裕，石尾 隆，井上克郎：ソースコード中に出現する動詞-目的語関係を利用したメソッド名の命名支援手法，信学技報，Vol.111, No.481, pp.1–6 (2012).
[10] Oracle Corporation: The java tutorial, available from ⟨http://java.sun.com/docs/books/tutorial/java/javaOO/index.html⟩.
[11] Microsoft: Method Naming Guidelines, available from ⟨http://msdn.microsoft.com/en-us/library/4df752aw(v=vs.71).aspx⟩.
[12] Høst, E.W. and Østvold, B.M.: Software Language Engineering, chapter The Java Programmer's Phrase Book, pp.322–341, Springer-Verlag, Berlin, Heidelberg (2009).
[13] Sun Microsystems: Java platform, standard edition 6 API specification, available from ⟨http://java.sun.com/javase/6/docs/api⟩.
[14] Fry, Z., Shepherd, D., Hill, E., Pollock, L. and Vijay-Shanker, K.: Analysing source code: Looking for useful verb-direct object pairs in all the right places, *IET Software*, Vol.2, No.1, pp.27–36 (2008).
[15] Miller, G.A.: WordNet: A lexical database for English, *Comm. ACM*, Vol.38, No.11, pp.39–41 (1995).
[16] Shepherd, D., Pollock, L. and Vijay-Shanker, K.: Towards supporting on-demand virtual remodularization using program graphs, *Proc. 5th International Conference on Aspect-Oriented Software Development*, pp.3–14 (2006).
[17] Hill, E., Pollock, L. and Vijay-Shanker, K.: Automatically capturing source code context of NL-queries for software maintenance and reuse, *Proc. 31st International Conference on Software Engineering*, pp.232–242 (2009).
[18] Høst, E.W. and Østvold, B.M.: The Programmer's Lexicon, Volume I: The Verbs, *Proc. 7th IEEE International Working Conference on Source Code Analysis and Manipulation*, pp.193–202 (2007).
[19] Zhou, H., Chen, F. and Yang, H.: Developing Application Specific Ontology for Program Comprehension by Combining Domain Ontology with Code Ontology, *Proc. 2008 The Eighth International Conference on Quality Software*, pp.225–234 (2008).

鹿島 悠

平成 22 年大阪大学基礎工学部情報科学科卒業．平成 24 年同大学大学院修士課程修了．同年同大学院博士課程入学．

早瀬 康裕 （正会員）

平成 14 年大阪大学基礎工学部情報科学科卒業．平成 19 年同大学大学院博士後期課程修了．同年同大学特任助教．平成 22 年東洋大学総合情報学部助教．平成 23 年筑波大学システム情報系助教．博士（情報科学）．オープンソースソフトウェア開発，ソフトウェア保守の研究に従事．IEEE 会員．

眞鍋 雄貴 （正会員）

平成 18 年大阪大学基礎工学部情報科学科退学．平成 23 年同大学大学院情報科学研究科博士後期課程修了．同年同研究科特任助教．博士（情報科学）．ソフトウェア再利用，ソフトウェアライセンスの研究に従事．ACM 会員．

井上 克郎 （フェロー）

昭和 59 年大阪大学大学院基礎工学研究科博士後期課程修了（工学博士）．同年大阪大学基礎工学部情報工学科助手．昭和 59～61 年ハワイ大学マノア校コンピュータサイエンス学科助教授．平成 3 年大阪大学基礎工学部助教授．平成 7 年同学部教授．平成 14 年大阪大学大学院情報科学研究科教授．平成 23 年 8 月より大阪大学大学院情報科学研究科研究科長．ソフトウェア工学，特にコードクローンやコード検索等のプログラム分析や再利用技術の研究に従事．