実装技術特定を目指した プログラムコード実装と説明的文書との対応付け調査

山下 大貴[1]　竹内 和広[1]

概要：従来，プログラムコードに含まれるメソッドあるいはコメントから，プログラムで使われる自然言語の語間の意味関係の同定を行い，それらの類型を示すことで，プログラム作成上の語間の関連性を明らかにする研究がなされてきた．従来研究では，語の関連性が，プログラミング実装技術にどのように関連するかは，明らかにされていない．この問題に対し，本研究は，ソースコードに含まれるクラス名，メソッド名，変数名等の識別子に使用される語間の関係を同定した上で，従来研究でなされてこなかった，ソースコードの開発目的となる対象を自然言語で記述した百科事典的な説明文書との対応付けに基づく調査を試みる．具体的には，遺伝的アルゴリズムをはじめとする複数のプロジェクトのソースコードにおける命名パターンと，それらのプロジェクトそれぞれの対象や目的に関する説明記事とを対応付ける分析を行った．その結果，複数プロジェクト間で共通する命名パターンが存在し，それら共通命名パターンには実装技法に関わる語が使用されていることを確認し，それらの語をより精緻なソースコードマイニングに応用できるように，辞書化を試みた．

Comparison of use of words between program code and the corresponding document to identify implementation pattern

Hiroki Yamashita[1]　Kazuhiro Takeuchi[1]

Abstract: There are not a few works on automatic identification of semantic relation of natural language words that appear in program codes have been conducted in recent years. Although these works clarified that there are differences between the word usage in program codes and that in general text such as newspaper, discussions about the explanation of those differences have not been made well. In this paper, we show that the word usage in program codes relates to programming implementation techniques from the comparison between the word usage of program codes and that of the explanatory articles that describe their target. To address the problem, we use documents in Wikipedia as explanatory articles and investigate three targets of program codes. We employ the Randomized SVD, which strengthened the general SVD with a sampling technique, to identify the semantic relation among the words in program codes such as the class names, the method names, the variables. As a result, we compiled the collected words that relate to program implementation into a dictionary for more accurate source code mining.

1. はじめに

ソースコードに含まれる自然言語には，コメントの記述のように通常の自然言語による記述が存在し，クラス名，メソッド名，変数名といった構成要素にも自然言語を利用した独特の記述がなされる [1][2]．そして，これらの要素がソフトウェア工学における設計思想，採用モデル，制作技術に強く相関すると考えられ，ソースコードの自動解析や品質評価等のより深いマイニングを行う場合には，自然言語により記述される要素に注目する必要があると思われる．このような要素を扱うには，プログラミング言語の予約語の数をはるかに超える数の自然言語の語を扱わなくてはいけない問題と，語用を考慮しなくてはならない問題が生じる．

[1] 大阪電気通信大学 大学院工学研究科
Osaka Electro-Communication University, Graduate School of Engineering

ソースコード上の識別子は，プログラム中で慣用的に使われる語，アプリケーション固有の語など，一般的英語辞書や WordNet[3] に記載される語には見られない専門的な語との組み合わせで命名され，一般的な語用とは異なる専門的な語用がなされている．しかし，それらの研究では，プログラム上の語用の特性がプログラミング実装技術にどのように関連するのかは明らかにしていない．そこで本稿では，ソースコードのより深い分析を行うことを目標に，ソースコード上の語の運用とプログラム実装技術との相関を明らかにする調査を試みる．

2. 本研究の位置づけ

近年，データマイニング手法の基盤ツールが整備されてきたことから，プログラムコード上の単語についてもデータマイニング手法を利用した分析がなされるようになっている．例えば，プログラムコードに含まれるメソッドあるいはコメントから，プログラムで使われる自然言語の語間の意味関係について自動同定を行い，それらの類型を示すことで，プログラム作成上の自然言語の運用特性を明らかにする研究がなされている [4][5].

Yang らの研究 [5] では，コメントやコードに含まれる語のコンテキストを比較し，2 つの語やフレーズが同じコンテキストで使われていれば構造的，意味的に関連していることを仮定し，意味的に関連する語関係を抽出する研究を行っている．この研究で使用された手法は，まず，与えられたソースコードに含まれる全てのメソッド，コメントを抽出し，それらを語のシーケンスに分割する．この得られたシーケンス間の類似度を計算し，与えられた閾値を越えた場合，2 つのコンテキストが類似しているとみなし，関連する語彙ペアとして抽出している．そして，抽出した語間の関係類型を示した上で，それらの関係付けが，一般的な語用と異なることを示している．この研究が整理した関係類型は次の通りである．

- synonym（同義語）
- related（関係している語）
- antonym（反意語）
- near antonym（反意語に近い語）

これに対して，プログラム実装上の自然言語における語用を分析し，ソースコードとのマッピングを検討したものに Howard らの研究 [6] がある．この研究では，メソッドに対する説明が記述されるコメントで用いられる動作動詞がメソッドの動作動詞と意味的に類似しているという仮定に基づいて意味的に類似する語のペアの抽出が行われている．また，メソッドに対しての説明コメントは，メソッドの振る舞いの全体的な概要を提供するものとして定義している．

この研究で用いられた手法は，まず非説明的であるコメントを除外し，次に，メソッドのシグネチャとメソッドに対応する説明的コメントから主要な動作語を抽出する．メソッドの主要な動作語は，メソッド本体の意図を表現する語彙であると定義している．そして，コメントとコードから得られた語のペアは，意味的に類似する語のペアの候補を形成するとして抽出を行っている．

ソースコードからマイニングされた語が，プログラム実装とどう関わるかに踏み込んだ研究に，柏原らの研究 [7] がある．具体的には，相関ルールマイニングを用い，メソッドとその名前を対応付ける命名相関ルールを抽出し，多くのソースコードで頻出する関係を抽出している．このとき，相関ルールマイニングで用いるトランザクションをメソッド単位で解析したメソッドのコンテキストの集合として表現し，メソッドのコンテキストは，1 つのメソッド周辺に出現した識別子との集合の組としており，命名相関ルールとして条件部がメソッド名の動詞・名詞以外からなるもの，帰結部にメソッド名の動詞のみが含まれているものとしている．また，この相関ルールの中から語間の共起性に基づき，述語と項の関係といった語用を命名パターンとして抽出している．

一般的に，プログラムには，それが用いられる目的があるはずである．例えば，Web ブラウザであれば，Web 上にある HTML ページをユーザが閲覧することが目的であり，Web ブラウザのソースコード全体は何らかの小目的を持つプログラム群が大目的のために組織化されたものである．開発者がプログラムを作成する場合，プログラムの実装目的を考える必要があり，プログラムにはその考えが反映されるはずである．ここで，以上の考えを図 1 に示す．図 1 は，プログラマは文書や頭の中にあるプログラムの目的を，それを実現する上で特定のアルゴリズムやモデルを採用して，プログラムを実装するが，その実装方法に関する情報は，プログラム上の語の使用に反映される様を示す．例えば，スタックを用いて特定の目的のプログラムを実装するならば，採用した実装方法であるスタック構造を反映して，プログラム上で push や pop といった語が使用されるという仮説である．我々は，そのような実装方法に関係する語を，実装機能語と呼ぶ．

プログラム作成の目的に従って開発者が目的を達成することに寄与するプログラムコードを作成すると考えれば，紹介した関連研究は，開発者の持つプログラム実装に関する知識を，プログラムコードの集合からマイニングしたものと考えられる．つまり，プログラムコード中に，一般的な文書を書くことと違った，語の意味関係や用法，命名パターンといった語用に特性があることが，関連研究で明らかにされてはいるものの，それらにはプログラム実装の技術に強く関わるものがあると我々は考える．

ここで，自然言語の語用の諸相について，用語を定義したい．柏原らが語間の共起性に基づき，抽出した語の関連性を命名パターンと一律に呼んでいることに対し，本稿で

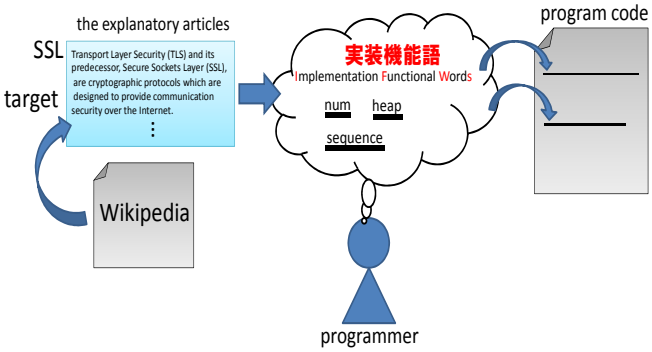

図 1　説明文書からプログラムコード記述に必要な知識
Fig. 1　Implicit knowledge behind writing the programming codes

表 1　実験データ
Table 1　List of the properties of the projets for our investigation

実装目的	プロジェクト名	行数	語彙数
遺伝的アルゴリズム	jgap	107417	1878
	EO	76002	2071
	beagle	83686	1614
	opt4j	43676	1276
ニューラルネットワーク	neuralnetworks	12755	543
	NeuralNetworkToolkit	19763	662
	neuroph	23926	881
	opennn	152873	1050
SSL	openssl	390534	7061
	cl342	362250	6017
	cyassl	102555	2936
	polarssl	85252	2191

は命名パターンをプログラムの実装対象中の語用と実装対象をプログラム上で実現する上での実装に結びついた命名パターンを区別したい．これらの語用を区別する基準として，分析対象のプログラムの開発目的となっている対象に関する自然言語で記述した説明文書である Wikipedia[8] を用いた．

一般的に，Wikipedia の説明文書は，個別のプログラム実装に関わる設計思想やモデルなど，具体的な内容は詳しく書かれていない．つまり，従来手法と同様な手法により抽出された命名パターンに対して，説明文書である Wikipedia 上にも出現する命名パターンと，そうではない語を客観的に分けることができる．そのとき，前者はプログラム対象の一般的な記述に関わるものであり，後者は，説明文章には書かれていない実装に関わる語用ではないかと考える．これを，同じ目的をもつ複数のプロジェクトの分析を行い，この調査を精緻にすることを本稿の残りで検討する．

3. 複数プロジェクトに対する調査

調査の対象となる複数プロジェクトとして，表 1 に示すデータを使用し，複数プロジェクトに関する命名パターンの調査の全体像を図 2 に沿って説明する．図 2 の右端のように，同一目的の複数プロジェクト A，B，C，D に対して，ソースコード専用の解析器を用いてプログラムコード内に含まれるクラス名，メソッド名，変数名等の識別子を抽出し，抽出した識別子から語を分割して抽出する．次に，個々のプロジェクトに対して関連語抽出 [9][10] を用いてプログラムコードに含まれる語間の意味関係を同定する．この結果から，個々のプロジェクトで得られた関連語集合に対し，その集合内で語のペアを生成し，これらの語のペアを複数プロジェクト間で比較することにより，複数のプロジェクトにおいて共通して見られる命名パターンを共通命名パターンとして抽出する．

この共通命名パターンに対して，パターンを構成する 2 つの語がプログラムの目的対象となる語か否かを調査する．具体的には，それぞれのプロジェクトの目的となる説明文書を Wikipedia から抽出し，その命名パターンの 2 語とも，あるいは，片方がそれらの説明文書中に存在するものを取り出す．このとき，Wikipedia 自体には，対象の実装に関わる設計思想やモデルなどは記述されていないことを考え，図 2 のように，抽出した命名パターンに対し，一方の語がソースコードと Wikipedia 上の語で共通し，他方が Wikipedia にはない語彙で構成される語のペアであれば，その語のペアが Wikipedia には見られない設計思想，モデルなどの隠された知識を含む命名パターンである可能性があるといった仮説に基づき，Wikipedia を基準とする実装機能語候補の抽出を試みる．

以上の処理は，汎用的なソースコードマイニングツールとして整備しており，そのツール群を用いて行った．プログラムコードはキャメルケース等の規則に従って語に分解され，Randomized SVD[11][12] を使った関連語集合への分類処理を行う．得られた関連語集合は，WordNet や Wikipedia といった言語資源と多角的に比較できる機能を持たせてある．なお，このツールは我々の Web ページで公開予定である．

4. 結果と検討

4.1 実装に関わる語の特定

対象プロジェクトに含まれるソースコードに記述される語に対し，Randomized SVD による次元圧縮による関連語集合への分類を行い，次元圧縮で得られた意味的関係をもつ任意のペアに対して，他の関連研究と同様にデータマイニングツールを使った語の意味の分析を行った．ただし，他の関連研究とは異なり，本研究では，意味的関係の発見基準となる次元数の決定に WordNet[3] を用いた客観的手法を用いており，抽出された語のペアの認定の際にも，客観的な基準として，Wikipedia の記述に出現する語関係と一致するか否かを用いた．また，主眼となる分析は類語関係の分類ではなく，実装技術に関わる命名パターンの分析である．

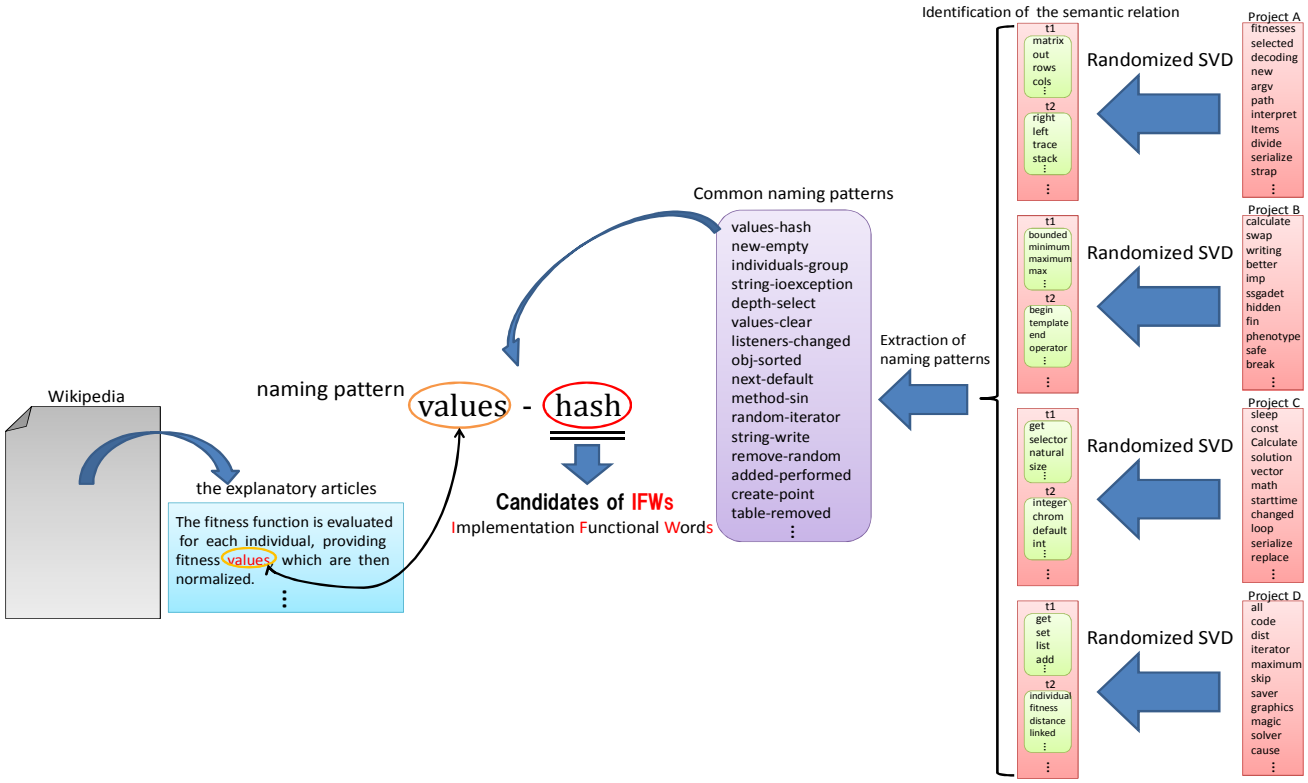

図 2 命名パターン抽出処理手順
Fig. 2 The outline of extracting semantic relations among the words

この調査により，3 つの目的をもつ，計 12 のプロジェクトに対して，同一目的間で共通して見られる命名パターンを，客観的な基準に基づいて分類した例を表 2 に示す．表中の同義語 (Synonym) の列は，関連研究が示してきた同義語関係を参考のために記した．また，調査対象とする語のペアのうち，Wikipedia には記載されていない語には下線を示した．表 2 に記載される各実装目的における命名パターンは，頻度を基準に上位から順に選択している．

表 2 の遺伝的アルゴリズムでは，genes-instance， new-instance のようにオブジェクトの生成に関わる命名パターンが得られている．我々が実装機能語と定義した語の特徴は，対象をプログラムで表現する上での実装に関わる語である．例えば，next-iterator といった語のペアにおいて，iterator はプロジェクトが対象とする遺伝的アルゴリズムに関する Wikipedia の説明記事上にはない語であり，客観的基準から実装技術に関係する命名パターンであると考えられる．実際に，iterator は，オブジェクト指向設計における GoF デザインパターン [13] の 1 つである Iterator パターンに関する語であり，より抽象的なデータ構造を反映している．これは実装技術の特定に，客観的基準として Wikipedia が有効に働いていることを示唆している．

また，表 2 のニューラルネットワークでは，客観的基準に基づき抽出した命名パターンとして value-double， list-arrays などが得られており，これらの語はプログラム上においてデータ型やデータ構造を表す語である．さらに，matrix-weighted や error-weight， training-rate 等の対象中の関係を抽出している．さらに，表 2 の SSL においても Wikipedia による客観的基準により，create-allocs などのメモリ割り当てに関する実装技術を表す実装機能語が得られており，sign-convert， send-msg， make-sequence が表すように SSL における通信制御・暗号化に関わる対象中の意味関係が抽出されている．

以上のように，客観的基準で抽出された命名パターンにおける Wikipeida 未出現語は，我々が仮定した実装機能語の候補として妥当のように思われる．しかし，各目的において共通する上位頻度の命名パターンという抽出方法では，より精緻な分析のための抽出が課題となる．また，抽出された実装機能語は分析のための共有資源として辞書化すべきであると考える．そこで，そのような実装機能語辞書構築のための予備的な検討を次節に示す．

4.2 実装機能語の辞書化に対する検討

前節において，プログラムの集合から抽出した語用に対して Wikipedia を基準として実装に関わる語用とプログラムが対象としている一般的な記述に関わる語用に分類した．

既に述べてきた実装機能語に対する仮説は次の通りである．ある対象における実装方法は，開発者により異なる．例えば，ナップサック問題において，プログラマはプログラム上で配列を用いた実装やリストで実装する．すなわち，ある対象のプログラム実装では，いかにそのプログラ

表 2 各実装目的から得られた命名パターン
Table 2 Categorizing the extracted semantic relations among the word pairs

実装目的	Synonym	命名パターンが Wikipedia 含有語のみ形成	命名パターン中の 1 語のみが Wikipedia 含有語	
遺伝的アルゴリズム	upper-over	individual-fitness	values-hash	random-argument
	empty-remove	crossover-points	genes-instance	bits-score
	genetic-individual	string-code	next-iterator	gene-allele
	property-parameters	genes-pool	table-removed	board-problem
	get-set	next-child	new-instance	distance-candidates
ニューラルネットワーク	find-found	training-rate	list-iterator	logistic-exp
	set-get	activation-functions	data-map	layer-condition
	layer-structure	neuron-multiple	input-load	statistics-deviation
	weights-bias	rate-momentum	value-double	matrix-weighted
	read-load	error-weight	list-arrays	neuron-transfer
SSL	check-verify	keys-verify	sign-convert	session-checksum
	signature-sign	client-request	include-inline	mach-address
	finished-done	system-storage	make-sequence	send-msg
	end-shutdown	session-ticket	heap-chain	encrypt-parse
	encrypt-encode	code-signature	create-allocs	address-ipv

ムが実装されたか何らかの知識の表出がプログラム上に存在するものと考えられる．その表出の一つが，命名規則における語の語用であり，我々はそれらの語を実装機能語と呼んだ．

この節では，前節で得られた実装機能語の候補をノイズが含まれることを含めて辞書を試験的に構築し，それら実装機能語が実装に関わる語として捉える仮説の妥当性と，また応用への可能性を確認・検討する．本節では，実装機能語は，命名パターンの一方が Wikipedia 上の記事に含まれない語という条件で抽出した語の集合である．厳密には，それらの語はあくまでも実装機能語の候補にすぎないが，表 1 のプロジェクトから抽出した 319 語を，ここではあえて実装機能語と考えて議論を進める．実装機能語辞書に含まれる語の一部を表 3 に示す．

検討のためのデータは，辞書の語を抽出したプロジェクトと同一目的の他プロジェクトと，目的の異なる他プロジェクトである．つまり，検討対象となるデータは，辞書構築には利用していない．具体的なプロジェクトは表 4 に示した．これらのプロジェクトは GitHub[14] にあり，各プロジェクトの評価として star が割り当てられている．star の数が多いほど有益であることを示す，すなわち，この star がプロジェクトの品質の善し悪しの客観的基準になると考え，実装機能語辞書を使って，ソースコードの品質を評価できるかを検討したい．検討の方法は，実装機能語辞書と評価対象となるプロジェクトから抽出した実装機能語を比較し，対象のプロジェクトのソースコード群が辞書内の実装機能語をどれほど含んでいるかを示す単純な含有率を用いる．

表 3 辞書に含まれる実装機能語の一例
Table 3 Examples of words included in the dictionary

実装機能語
init, string, file, max, iterator, row, save, get
listener, remove, release, upper, num, flush, index
matrix, label, new, object, run, put, path, column
load, array, total, val, add, params, prev, empty
flag, true, count, char, default, compare, parse
illegal, min, write, instance, ioexception, map, trace
clone, update, log, pair, pool, dummy, comment
assert, text, child, results, weights, certs, stack

実験の結果を表 4 に示す．表 4 が示すように，未知と既知のプロジェクトの評価指標である star と実装機能語の含有率に対し，star 数が大きい場合に実装機能語の含有率が高くなり，star 数が小さい場合に語の含有率が低くなっている．すなわち，辞書を用いた単純な計量を用いても，プロジェクトの品質の指標と何らかの相関があることを確認できる．この表 4 の実験は，データ数が限られており，また，試作的な辞書を使っているため，より精密な統計的な分析と検証は今後の課題となるが，構築した辞書に含まれる実装機能語がプログラムの対象に関わる実装の善し悪しを判断することに強く貢献することが期待できる．この結果は，命名規則に数多く含まれる実装機能語が，頑健な品質のプログラムを作成する上でのプログラムに潜在的に共通して存在している実装知識と強く関係することを示唆しており，我々の手法により構築した実装機能語辞書が，プログラムが目的としているアルゴリズム・手法のプログラム実装に関わる語用を抽出できたものと考えている．

表 4　検証データ
Table 4　The properties of the data for evaluation of the extracted dictionary

実装目的	プロジェクト名	行数	語彙数	star	A:辞書内の語と一致した数	A/辞書内の語数 (=319)
ニューラルネットワーク	caffe	48031	1913	715	182	0.57
	nnForge	39237	918	76	101	0.32
	jnn	6938	344	26	67	0.21
	NeuralNetwork	1856	287	9	58	0.18
DNS	gdnsd	23885	2059	72	129	0.41
	robdns	23251	1286	26	109	0.34
	Netty-DNS-Server	10659	801	10	104	0.32
	dnsjava	35278	1376	6	100	0.31

5. おわりに

本稿ではプログラムコード上の語用の特性が，プログラミング実装技術にどのように関連するかを明らかにする調査を行った．具体的には，柏原ら [7] の手法や他の研究での語の共起性に基づく語の関連性分析に対して，本稿の特色は，Wikipedia の説明記事との対応付けを行うことで，Wikipedia の説明記事に掲載されている対象を記述する語用と，プログラムの目的を達成するために選択されたアルゴリズムや実装モデル・技法に関わる語用とを区別して分析・検討したことである．具体的には，Howard ら [6]，Yang ら [5] が明らかにしていない，synonym や antonym といった意味関係とは異なる種類の語用として，Wikipedia に存在しない語が対象となるアルゴリズムや技法における開発者の設計思想やモデルといった実装に関係する語関係を抽出していることを明らかにした．

さらに，得られた命名パターンからから実装機能語の候補を抽出し，実装機能語辞書として試作的に整理し，それを用いて未知のソースコードの品質評価を実験し，そこでの関係性を検討することにより，辞書の応用の可能性とより精緻なソースコード分析の可能性について議論した．本研究で得られた知見は，ソースコード中の命名規則から，実装のためのモデルや考え方，パターンという抽象度の高い部分に対する，より精緻なソースコードに対するマイニング技術の基盤構築に貢献できるものと考える．

今後は，この種のマイニング技術の基盤として，実施機能語辞書をより精緻なものにすると共に，辞書内のそれぞれの語が具体的にどのような実装モデルに基づいてプログラムを作成する場合に使用されるかを整理していく予定である．また，説明文書として利用した Wikipedia 記事をより精密に構造化して利用すると共に，Wikipedia 記事以外の仕様書等のプログラム実装に直接関わる文書との対応付けも検討していきたい．

謝辞

この研究の一部は科研費 (基盤 (C) 課題番号:24501158) の支援を受けている．

参考文献

[1] E. W. Høst, and B. M. Østvold, The Programmer's Lexicon, Volume I: The Verbs, Proceedings of the Seventh IEEE International Working Conference on Source Code Analysis and Manipulation, pp.193-202 (2007)
[2] E. W. Høst, and B. M. Østvold, Debugging Method Names, Proceedings of the 23rd European Conference on ECOOP 2009 — Object-Oriented Programming, pp.294-317 (2009)
[3] WordNet, http://wordnet.princeton.edu.
[4] D. Shepherd, Z. P. Fry, E. Hill, L. Pollock, and K. Vilay-Shanker, Using Natural Language Program Analysis to Locate and Understand Action-oriented Concerns, Proceedings of the 6th International Conference on Aspect-oriented Software Development, pp.212-224 (2007)
[5] J. Yang and L. Tan, Inferring Semantically Related Words from Software Context, Mining Software Repositories (MSR), 2012 9th IEEE Working Conference on, pp.161-170 (2012)
[6] M. J. Howard, S. Gupta, L. Pollock, and K. Vijay-Shanker, Automatically Mining Software-Based, Semantically-Similar Words from Comment-Code Mappings, Proceeding MSR '13 Proceedings of the 10th Working Conference on Mining Software Repositories, pp.377-386 (2013)
[7] 柏原由紀, 鬼塚勇弥, 石尾隆, 早瀬康裕, 山本 哲男, 井上克郎: 相関ルールマイニングを用いたメソッドの命名方法の分析, 日本ソフトウェア科学会, ソフトウェア工学の基礎 XX, pp.25-34 (2013)
[8] Wikipedia, http://www.wikipedia.org/
[9] A. Kuhn, S. Ducasse, and T. Gírba, Semantic Clustering: Identifying Topics in Source Code, Information and Software Technology 49, pp.230-243 (2007)
[10] C. D. Manning, P. Raghavan, and H. Schutze, Introduction to Information Retrieval, Cambridge University Press (2008)
[11] 岡野原 大輔: 全部分文字列のクラスタリングとその応用, 言語処理学会 第 17 回年次大会 発表論文集 (2011)
[12] N. Halko, P. G. Martinsson, and J. Tropp, Finding Structure with Randomness: Probabilistic Algorithms for Constructing Approximate Matrix Decompositions, Society for Industrial and Applied Mathematics, SIAM REVIEW Vol.53, No.2, pp.217-288 (2011)
[13] E. Gamma, R. Helm, R. Johnson, and J. Vlissides, Design Patterns: Elements of Reusable Object-Oriented Software, Addision Wesley Professional (1994)
[14] GitHub, http://github.com