メソッド名とドキュメンテーションコメントの対応付け手法の提案

A Method to Find Corresponding Terms in Documentation Comments to a Method Name

加古径吾*　大久保弘崇†　粕谷英人‡　山本晋一郎§　稲垣康善¶

あらまし　Java のメソッド名に使われる単語と，コメント文で使われる単語の対応付けを行う手法について述べる．メソッド名とそのコメント文の双方を構文解析し，係り受け構造の対応をとることで，語の省略，言い替えのパターンを既存のプログラムから抽出する手法を提案する．この情報を用いれば，新たなメソッドを作成する時，自然言語文による説明から，パターンに従って変換を行えばプロジェクトの慣習に沿ったメソッド名を決定するための指針になる．

1　はじめに

Java のメソッド名は，英単語の動詞，名詞，形容詞といった組み合わせによって記述されている．開発者はメソッド名を構成する単語を見ることで，そのメソッドがどのような役割を持つか推測することができる．しかし，英語には類義語が存在するため，ある役割に対する表現が人によって異なってしまうことがある．そして，本来は同じ役割を持つメソッドでも，それぞれ違う単語を使って命名してしまうことが起こる．

Java にはコーディング規約 [1] が存在し，メソッド名の生成規則が規定されている．そして，規約に準拠したメソッド名を定義することで，開発者間でメソッド名のスタイルを統一することができる．だが，これだけでは前に述べたような類義語の問題を解決することができず，使用する単語の曖昧さを解決することができない．

このように，役割によって使用する単語を明文化しないでメソッドを命名してしまうと，プロジェクトのメソッド名に一貫性がなくなり，可読性が低下し，バグの原因ともなりうる．

本研究は，この問題に対処するために，プロジェクトの既存のソースプログラムに存在するメソッド名を利用して，使用する単語の統一性が崩れないように新しいメソッド名を決定する支援を目的とする．

メソッド名で使われている単語は，そのメソッド名に対応するコメント文での単語や表現が置き換えられたものと考えられる．新たなメソッド名を決定するときに，この置き換え規則を利用することで，そのプロジェクトの役割の表現方法に沿う，メソッド名候補を提示する方法について考える．

本稿では，構文解析を利用したコメント文とメソッド名の単語対応付け手法を提案し，その評価結果を報告する．

2　対象とするメソッド名とコメント文

本稿で対象としているメソッド名は，コーディング規約に準拠したものとしている．具体的には以下の規則を守ったメソッド名である．

*Keigo Kako, 愛知県立大学大学院 情報科学研究科, kako@yamamoto.ist.aichi-pu.ac.jp

†Hirotaka Ohkubo, 愛知県立大学 情報科学部, ohkubo@ist.aichi-pu.ac.jp

‡Hideto Kasuya, 愛知県立大学 情報科学部, kasuya@ist.aichi-pu.ac.jp

§Shinichiro Yamamoto, 愛知県立大学 情報科学部, yamamoto@ist.aichi-pu.ac.jp

¶Yasuyoshi Inagaki, 愛知工業大学 経営情報科学部, y-inagaki@aitech.ac.jp

1. useLocalInvocation
2. getLeftSideBearing
3. length
4. isEmpty
5. toArray

図 1　規約に準拠したメソッド名

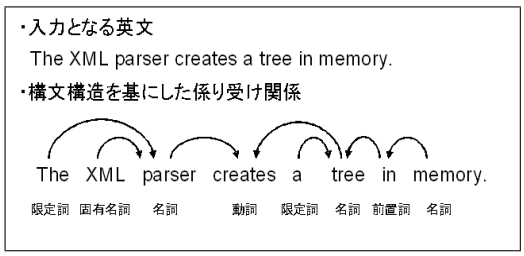

図 2　得られる構文構造

1. メソッド名は動詞または動詞句ではじまり，最初の文字は小文字に，続く単語は最初の一文字を大文字とする．
2. 変数 X に対する属性を取得 (get)，設定 (set) するためのメソッド名は，getX や setX とする．
3. 何かの長さを返すメソッド名は，length とする．
4. オブジェクトに関する条件 X の真偽値 (boolean) を判定するメソッド名は，isX とする．
5. オブジェクトを特定の形式 X へ変換するメソッド名は toX とする．

上記の規則に従うメソッド名の例を図 1 に挙げる．

コメント文は，同じくコーディング規約に準拠しているものとし，メソッド名に対応するドキュメンテーションコメントの最初の一文を対象とする．この文は，規約によりメソッドの役割に関して簡潔な文 (要約文) を書くよう定められている．そのため，要約文で使用される単語がメソッド名の中の単語として，同一か置き換えられて登場する可能性が高いと考えた．このように，規約が守られているプロジェクトとして，JDK1.4 [3] を実験に用いた．

3　メソッド名とコメント文の構文解析

構文解析器には，Charniak Parser(以下 Charniak) [4] を使用する．英文を与えると，図 2 に示すように，単語の品詞と単語間の係り受け関係が得られる．矢印が係り受け関係を表しており，始点が係る側，終点が係られる側である．

3.1　メソッド名の構文解析

メソッド名を英文として Charniak で解析させるには，単語間を分かち書きし，英文の形に近づける必要がある．分かち書きしたメソッド名にピリオドを付加することで Charniak による解析を行うことができる．しかし，メソッド名の品詞決定は主語を付加することで，Charniak の解析精度を上げることができるとわかった．そのため，JDK から重複なしのランダムに選んだ 100 個のメソッドを対象に，人称毎の主語を付加し解析した結果，正しく品詞付けされたメソッドが一番多い主語を調べた．その結果，‘I’，“You”，“It”，“This” の中では，“I” が正しい品詞付けを一番多くしていることが分かった．したがって，各メソッド名には ‘I’ の主語を付加することで，構文解析を行うことにした．以上の処理を行うと，メソッド名は，図 3 のように変形される．生成規則の 4 と 5 に従うメソッド名は，それぞれ英文が正しくなるように “I” の代わりに “It” と “The value changes” を付加する．

3.2　コメント文の構文解析

コメント文は，図 4 のように，“/**” から “*/” までのドキュメンテーションコメント全体を抽出し，その中から対象としている要約文を抽出する．

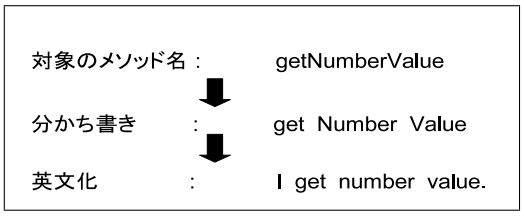

図 3　構文解析に使用するメソッド

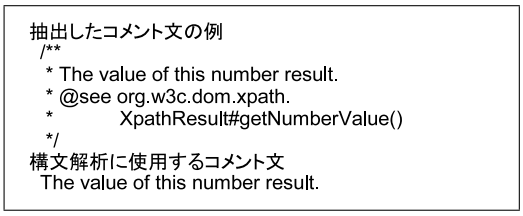

図 4　コメント文の要約文の特定

3.2.1　最初の文の特定

英文で書かれたコメント文を対象としているため，最初にピリオドが出てきた所までを抽出する．また，改行を取り除き一行の文字列にする．そして，ドキュメンテーションコメントの記法である，‘/’ と ‘*’ を文から削除する．最初の文の特定で抽出した文中に，タグや ‘@’ が含まれているとき，その部分は削除する．また，ドキュメンテーションコメント中に要約文が存在しないものは，メソッドに対する説明文が無いものとして解析対象からはずした．文中に人名，URL やパッケージ名等の一部としてピリオドが存在するものについては，判別しておらずそこまでを英文として抽出してしまっている．図 4 に要約文の箇所を取り出した例を示す．

3.2.2　主語の付加

JDK では要約文の半数以上が主語を省略しているコメントだったため，精度を上げるためにメソッド名と同様に主語を付加した．主語が省略されているコメント文の動詞の大半は，三単元の現在形であったため，“This method” を主語として付加した．ただし，抽出した文の最初の単語が “It” や “We” のような代名詞，“The” や “a” 等の冠詞で始まる時は，主語があるものとして新たに主語を付加しないようにした．

4　構文解析結果による単語間の対応付け

3 章の手順で得られたメソッド名とコメント文の構文解析結果を利用して，両者の単語の対応付けを行う．

単語間の対応には，同一の単語によるものと異なる単語同士によるものが考えられる．前者は，コメント文の中に，メソッド名で使われている単語が存在するときのことをさす．後者は，異なる単語同士でも同じ役割を表すために使われているときのことをさす．このような単語の対応付けの方法を以下に述べる．なお，3 節の処理過程で付加した単語は本来存在する単語ではないので対応付けを行わない．

4.1　同一単語の対応

メソッド名で使われている単語がコメント文にも出現するか検索する．コメント文では語形変化を考慮する必要がある．そのため，染谷らによる単語の語形変化辞書ファイル [5] を利用した．コメント文側に一致する単語があれば，両者を対応付ける．

4.2　係り受け関係を利用した異なる単語との対応

メソッド名で使われている単語がコメント文に存在せず，コメント文で別の表現で表されているとき，両者の構文構造から，どの単語で置き換えられているか対応付ける．以下にその対応付けの手法を述べる．

4.2.1　注目する単語に係る単語を利用した対応付け

メソッド名の単語がコメント文には存在せず，係り受け関係で係ってきている単語がある場合の対応付け手法を以下に示す．

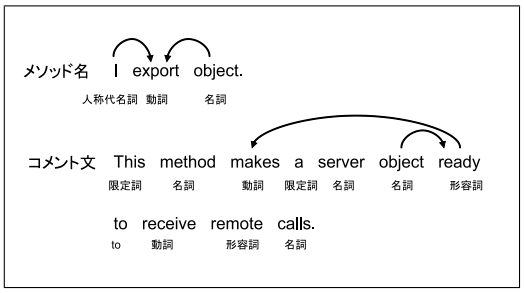

図 5　係る単語がある場合

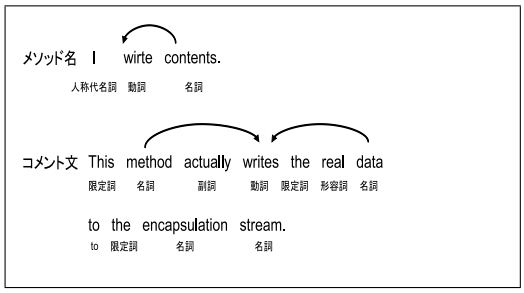

図 6　係っていく単語がある場合

手順 1　メソッド名の構文構造で，注目している単語 A に係っている単語 B を選ぶ．
手順 2　単語 B が，コメント文側にも出現している (C) なら次に進む．
手順 3　コメント文中の単語 C が，構文構造で係っている単語 D について，その品詞が単語 A の品詞と同じなら次に進む．
手順 4　単語 A が置き換えられた単語 D として対応づける．同じ品詞の単語 D が見つかるか係り受け関係がなくなるまで辿っていく．

この方法により，コメント文で使われる単語がメソッド名ではどのような単語に置き換えられるか見つけ出すことができる．

この手順で対応付けた例を図 5 に示す．なお，図 5 の係り受け関係には省略がある． 単語 “export” はコメント文には出現しない．そこで，係り受け関係を持っている，“object” に注目する．“object” は，コメント文側にも存在し，係る単語が存在する．係る単語 “ready” は形容詞であり，動詞である “export” と一致しない．そのため，さらに “ready” の係る単語を探索する．“ready” が係る単語 “makes” は，動詞の現在分詞であるので，“export” と品詞が一致する．したがって，“export” と “makes” を対応付ける．

4.2.2　注目する単語が係っていく単語を利用した対応付け

係る側と同様，メソッド名で注目している単語がコメント文に存在せず，係り受け関係で矢印先の単語がある場合の対応付け手法を以下に示す．

手順 1　メソッド名の構文構造で，注目している単語 A が係っている単語 B を選ぶ．
手順 2　単語 B が，コメント側にも出現している (C) なら次に進む．
手順 3　コメント文中の単語 C に，構文構造で係ってくる単語 D について，その品詞が単語 A の品詞と同じなら次に進む．
手順 4　単語 A が置き換えられた単語 D として対応付ける．同じ品詞の単語 D が見つかるか係り受け関係がなくなるまで辿っていく．

この方法により，コメント文で使われる単語がメソッド名ではどのような単語に置き換えられるか見つけ出す．

この手法で対応付けた例を，図 6 に示す．メソッド名の “contents” に対応する単語をこの手法により導く．この例でも，解析結果の中で必要となる係り受け関係だけを示す．図 6 では，構文解析の結果 “contents” が “write” へ係っていくことがわかっている．“write” はコメント文側で “writes” として存在し，“method”，“data” が係ってきていることがわかる．これらの単語の中で，“method” は 3.2.2 で構文解析のために付加した単語であるので除外する．残る “data” は “contents” と品詞が同じであるので，これを対応付ける．

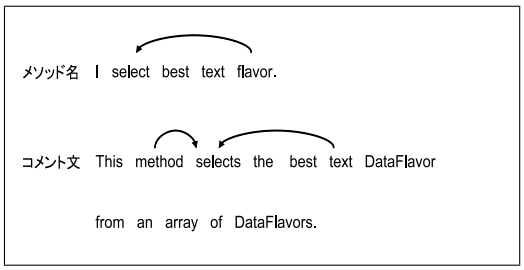

図 7　誤った対応付け

表 1　実験結果

パッケージ	メソッド数	適合率	再現率	F 値
java.applet	46	0.92 (72/78)	0.81 (72/89)	0.86
java.awt.datatransfer	83	0.84 (148/177)	0.76 (148/195)	0.80
org.w3c.dom	99	0.85 (159/187)	0.79 (160/203)	0.82
合計	228	0.86 (379/442)	0.78 (380/487)	0.81

5　対応付け手法の適用実験と評価

前章で行ったメソッド名とコメント文の単語間の対応付けは，同一の単語である場合と，別の単語である場合に分かれる．それぞれの対応付けが正しいかどうかの実験を行った．

5.1　判断基準

同一の単語の対応付け

対応付けられたものは正解とし，語形変化が辞書になかったものを不正解とみなす．

異なる単語との対応付け

対応付けた単語が，同じ表現であるか機械が判断することは難しいため，正しいかどうかは主観で判断する．

5.1.1　正解例と不正解例

図 5，図 6 は別単語の対応付けの正解例である．一方，不正解と判断した例を図 7 に示す．図では，メソッド側の “flavor” は “select” に係っており，コメント文では “select” に係っている単語は “method” と “text” である．“flavor” と “text” は名詞であり，置き換えられた単語として対応付けられる．しかし，実際には “flavor” がコメント文側で “DataFlavor” を表していることは明らかなので，これは対応付け失敗となる．

5.2　実験と評価

評価には，適合率，再現率，F 値を用いた．JDK のいくつかのパッケージに対し，メソッド名とコメント文の対の単語の対応付けを行った．表 1 は，同一の単語と別の単語での対応結果で正しいと判断したものの結果で，表 2 は，同一の単語の対応を除外し，別の単語のみで対応が正しいと判断した結果である．

同一単語の対応付けは，語形変化が辞書ファイルにあれば対応付けることができる．実験の評価に用いたメソッド名の中にはそのような単語が存在せず，全てを対応付けることができた．一方，別の単語間の対応付け結果に関しては半分程しか対応付けれていないことが 2 からわかる．

表 2　同一単語を除外した場合の精度

パッケージ	適合率	再現率	F 値
java.applet	0.76 (19/25)	0.53 (19/36)	0.62
java.awt.datatransfer	0.48 (27/56)	0.36 (27/74)	0.41
org.w3c.dom	0.61 (43/71)	0.51 (44/87)	0.56
合計	0.59 (89/152)	0.46 (90/197)	0.48

6　おわりに

本研究では，プロジェクトの慣習に沿った命名を行えるように，既存のプログラムを利用することで支援することを目的とした．そのために，メソッド名が英語の構文構造をとっている点に着目し，メソッド名と，それに対応するコメントの英文としての構文構造から，単語を対応付ける手法を提案した．そのため，メソッド名とドキュメンテーションコメントの要約文を対象に構文解析を行い，両者に登場する単語について対応付けを行った．その結果，同一の単語は辞書にないもの以外は対応付けを行うことができたが，別の単語として登場する場合，半分程しか対応付けられないことがわかった．そのため，現在の手法を見直し，別の単語同士でも，より高い精度で対応付けられるようにする必要がある．そして，その結果を基に，プロジェクトでのメソッド命名を支援することを今後の課題とする．

参考文献

[1] Code Conventions for the Java Programming Language,
http://java.sun.com/docs/codeconv/
[2] How to Write Doc Comments for the Javadoc Tool,
http://java.sun.com/j2se/javadoc/writingdoccomments/
[3] Java 2 SDK, Standard Edition,v 1.4.2,
http://java.sun.com/j2se/1.4.2/
[4] Eugene Charniak's ,
http://www.cs.brown.edu/people/ec/
[5] レマタイザ,
http://www.eng.ritsumei.ac.jp/asao/resources/lemma/