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.

コメント文によるプログラム中に出現する名詞の説明文生成

藤木　哲也†　　早瀬　康裕††　　井上　克郎†

† 大阪大学大学院情報科学研究科　〒 565-0871　大阪府吹田市山田丘 1 番 5 号
†† 東洋大学総合情報学部　〒 350-8585　埼玉県川越市鯨井 2100
E-mail: †{tetuya-f,inoue}@ist.osaka-u.ac.jp, ††hayase@toyo.jp

あらまし　ソフトウェア開発者はプログラム理解の際に，ソフトウェア中の識別子から関数や変数の役割や振舞いを類推する．識別子名中の単語の意味や用法は自然言語とは異なる場合があるために，作業者は類推を行うためにソフトウェア開発の経験からソフトウェア独特の意味や用法を学ばなければならなかった．本稿では，類推を支援するために，識別子名に出現する名詞の説明文を自動的に生成する手法を提案する．名詞の説明文の生成には，ソースコード中に記述されたコメントを利用する．コメントに対して自然言語処理を行い，名詞の説明を行っている箇所を抽出することで説明文生成を行う．また，実際に提案手法を適用した結果についても示す．
キーワード　プログラム理解，識別子，自然言語処理

Generating Descriptions of Nouns in Software from Program Comments

Fujiki TETSUYA†, Yasuhiro HAYASE††, and Katsuro INOUE†

† Graduate School of Information Science and Technology, Osaka University　Yamadaoka 1–5, Suita-shi, Osaka, 567-0871 Japan
†† Faculty of Information Science and Arts, Toyo University　Kujirai 2100, Kawagoe-shi,Saitama 565–0456 Japan
E-mail: †{tetuya-f,inoue}@ist.osaka-u.ac.jp, ††hayase@toyo.jp

Abstract　Software developers surmise roles and workings of functions or variables for program comprehension judging from its identifiers. It is difficult to surmise appropriately if engineers have little knowledge and experiment of software development, because sometimes word's meanings in identifiers are different form in natural language. This paper proposes a method to generate descriptions of nouns which are used in identifiers automatically that helps surmising identifiers. The description is generated from comments in source code by using natural language process and extracting notes for noun.
Key words　program comprehension, identifiers, natural language process

1. はじめに

近年，ソフトウェアの巨大化に伴って，ソフトウェア保守に要するコストの増大が問題となっている．ソフトウェア保守に要するコストのうち，プログラム理解にかかるコストが大きな割合を占めることが知られている [1], [9].

プログラム理解にとってドキュメントは有用な手がかりであるが，ドキュメントが存在しない場合や不十分な場合には，作業者はソースコードの読解により多くの時間を割きプログラム理解を行わなければならない．ソースコードの読解の際，ソースコード中の識別子は重要な手がかりとなる．識別子名から関数や変数などのプログラム要素の役割を類推しソフトウェア知識対応付けることで，ソースコードの読解を進める [8], [12].

識別子名は複数の単語から構成されることが多く，単語の組み合わせにより識別子の役割や振る舞いを表現している．そのため，識別子名に用いられる単語のそれぞれが，識別子のもつ構造や対象など振る舞いの一部を表している．

作業者が識別子名から識別子の役割や振る舞いを類推できない場合，プログラム理解に要する時間が増加する．ソフトウェアの開発経験やアプリケーションドメインの知識が不足している場合，作業者は識別子からプログラム要素の役割の類推とソフトウェア知識への対応付けを行うことができない．また，対応付けを行えない場合では行える場合に比べてプログラム理解に膨大な時間を必要とする．

作業者が識別子から類推を行えるようになるためには，実際の様々な事例から学習をすることが必要である．しかし，事例

からの学習には，多くのまた多様なドメインのソフトウェアの開発や保守を行わなければならず，学習に要するコストは大きい．そこで，私達の研究グループではソフトウェアに関する辞書を作成することでプログラム理解支援や識別子への適切な命名の支援を目指している [13], [14].

本稿では，識別子からの類推支援を目的とした識別子に出現する名詞の説明文を自動的に作成する手法を提案する．ソースコード中の識別子に対して記述されたコメントから，名詞を説明しているフレーズを抽出することで名詞の説明文を生成する．

以降，２節では研究背景について，３節では手法のアイデアを述べる．また，４節では手法のオブジェクト指向プログラムにおける前提を，５節では提案手法について，６節では提案手法の適用結果を示し考察を述べる．次いで，７節では関連研究を述べ，そして最後に８節で本手法のまとめと今後の課題を述べる．

2. 研究の背景

プログラム理解を行うために作業者がソースコードの読解を行うことは多い．プログラム理解を行うにあたって，ソフトウェアに付随するドキュメントは重要な手がかりである．しかし，ドキュメントはソフトウェアの変化の過程で適切に変更されないことや，紛失してしまうといったことがあるため，ソースコードの読解によりプログラム理解を行うことは多い．

作業者はソースコードの読解を行う際に，ソースコード中の識別子，コメント，返り値，関数の呼び出し関係などを手がかりとするが，その中でも識別子は非常に重要な手がかりの１つである．識別子名からその識別子の振る舞いや役割を類推することで，その識別子に関する箇所を読むべきかどうかを判断したり，プログラムの動作にある程度の予測をもって読み進めたりすることができる．ところが，作業者の知識や経験が不足し，識別子からの類推を適切に行えない場合には読解に要する時間が増加するという問題がある．関連の低い箇所の読解を行ってしまうことや誤った理解を持ったまま読解を進め手戻りが発生することがあるからである．

さらに，識別子からの適切な類推を行うための学習コストは高いという問題がある．その理由として，作業者が多くのソフトウェア開発に携わり事例から学ぶ必要があること，また，特定のドメインのソフトウェア中のみで使用される単語が存在するため多様なドメインのソフトウェアに触れることも必要であることが挙げられる．作業者は識別子に関する学習をソースコード，情報科学に関する教科書，仕様書や用語集などのドキュメントから学習を行う．ソースコードからの学習では，識別子とそれに対するコメントや識別子に関連する処理などから，識別子名とそれから類推される内容の対応付けを学ぶ．

名詞の説明文を作成することにより，作業者のプログラム理解を支援できる場面は多い．識別子名の多くには名詞が含まれており，また，識別子名に用いられる名詞の種類は膨大である．そのため，作業者はソースコードの読解の多くの場面で識別子中の名詞を目にし，名詞の説明文を収録した辞書を活用することができる．

3. 手法のアイデア

本手法では，ソースコード中のコメントを利用することで，識別子中に出現する名詞に対する説明文の作成を行う．

コメントにはライセンス情報，処理内容，開発者のメモなどプログラムに関することが記述される．そして，ある識別子に対して記述されたコメントでは，その識別子の処理内容や用途の説明が記述されている．そのような識別子への説明の中には，識別子名に含まれる単語の説明を行っている箇所があり，その記述を抽出することで名詞の説明文を作成する．

説明文の抽出は文よりも小さいフレーズ単位で行う．文単位での抽出では名詞の説明とは関係のない識別子全体への説明なども含まれてしまうことがある．自然言語文から重要な情報を取り出す方法として自動要約があるが，既存の自動要約手法では文ごとの重要度を評価し文単位の抽出を行うために説明文の作成には適さない．

本手法では，名詞への説明の抽出をフレーズ単位で行う，そのために複数のコメント中である名詞に対して共通してなされている説明箇所を抽出することで行う．

4. オブジェクト指向プログラムにおける識別子とコメント

本節では，オブジェクト指向プログラム中の識別子とコメントに関する規則について述べる．

4.1 複合語の表記法

識別子に用いられる複合語では複数の単語組み合わせるために，各単語を連結しひと繋がりの単語として表記する場合がある．一般に，単語の連結のパターンとして CamelCase と snake_case が用いられる．CamelCase では各単語の先頭を大文字にして，snake_case では単語境界にアンダースコアを挿入して連結を行う．

4.2 コメントの記述

コメントは識別子の直前もしくは直後に記述される．ソースコード中の任意の場所にコメントを記述できるが，一般的にはコメントの内容と関連する箇所に記述される．識別子に対して記述されたコメントはその識別子の直前に記述されることが多い，また，短いコメントであれば識別子の直後記述される場合もある．さらに，開発者はドキュメンテーションコメントを利用することで，どの識別子に対するコメントであるかを明示することもできる．

コメントに記述される内容としては，識別子の処理内容の説明など以外にもアノテーションを利用した作成者，作成日，ソフトウェアのバージョン情報や，他にもリファレンスを自動生成するための HTML タグなどがある．

5. 提 案 手 法

提案手法の概要を図１に示す．

本手法の入力は Java で書かれたソースコード集合であり，出力は識別子中に出現する名詞とそれに対する説明文との組である．

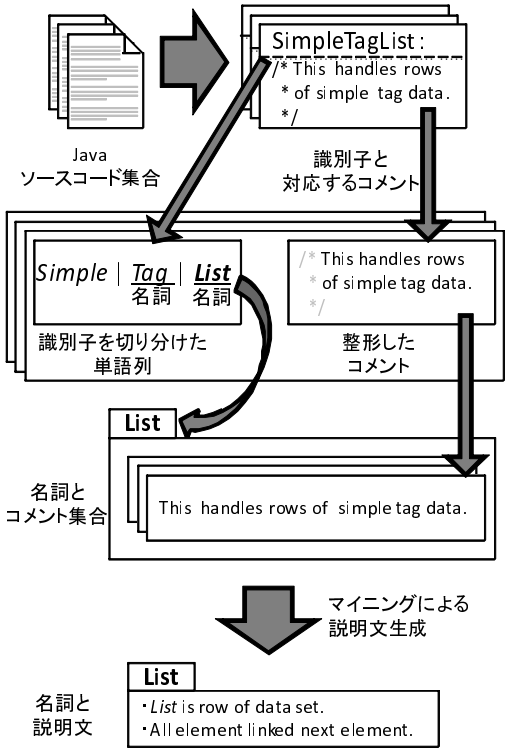

図 1　提案手法の概要
Fig. 1　abstract

提案手法ではソースコード集合を構文解析し，名詞とそれに対するコメント集合の組を作成する．それから，コメント集合に繰り返し出現する表現をその名詞への説明とし抽出することで説明文を生成する．

動作は以下の４つのステップからなる．

ステップ１ 識別子-コメントの組収集： ソースコード集合から，識別子の宣言とそれに付随するコメントを取得する．

ステップ２ 名詞-コメント集合の組作成： 取得した識別子から名詞の抽出とコメントの整形を行う．

ステップ３ 頻出する説明の抽出： 収集したコメント中に頻出する説明をコメントのグラフ化とグラフマイニングを用いて抽出する．

ステップ４ 説明文の生成： グラフマイニングの結果を自然言語の文に復元することで，説明文を生成する．

5.1　識別子-コメントの組収集

入力されたソースコード集合に対し構文解析を行い，各クラス名，およびコメントを抽出する．クラス宣言の直前に記述された１つのブロックコメントもしくは連続するラインコメントを，そのクラスに対応するコメントとし，識別子-コメントの組として抽出する．

5.2　名詞-コメント集合の組作成

ソースコード集合の解析から得られた識別子-コメントの組を用いて，識別子に含まれる名詞の抽出と，コメントの整形から説明を行うコメント文を抽出する．その後，名詞-コメント集合の組を作成する．コメント集合に同じコメントが複数存在すると，グラフマイニングの結果でそのコメント全文が抽出されしまうため，コメントの重複を取り除く．

5.2.1　識別子から名詞抽出

識別子が複合語であるなら，単語の連結パターンに則って識別子を単語に切り分ける，また各単語の品詞解析を行い，名詞である単語を抽出する．

5.2.2　コメントの整形

本手法では自然言語による説明を利用するので，コメント中から他の情報を除去することで説明のみを取り出す．取得したコメントから以下の記述を除去する．

コメント区切り文字： //,/***/

HTML タグ： <> で囲まれる領域

アノテーション： ブロックタグ（スタンドアロンタグ），インラインタグ

次に，コメントでは文の区切りが明確でない場合があるため，改行の修正を行う．次の行の先頭が小文字であった場合，改行を取り除き２つの文を結合する．文中でピリオド，感嘆符や疑問符 (文の区切り文字) が出現した場合，その直後で改行する (識別子等のカスケードは除く)．

5.3　コメントからのマイニング

コメント集合中に頻出するフレーズを抽出する．コメント文には特定のクラスに対する説明など単語と関係のない説明が含まれているので，文よりも小さい単位での抽出を行う．そのために，コメント文を構文解析し，文の構造をグラフとして表現する．その後，グラフ群に対してグラフマイニングを行うことで部分グラフの抽出を行う．

5.3.1　コメント文のグラフ化

まずコメント文を構文解析することで，文中の単語間の依存関係を取得する．構文解析には HPSG 理論に基づいた文法と高速な解析アルゴリズムを利用した英語構文解析器 Enju [7] を利用する．構文解析により文内の単語間の修飾被修飾関係と各単語の正規化を行った単語を取得する．単語の正規化とは，複数形の名詞を単数形に，過去分詞の動詞を原形などに変化させることである．次に，正規化された単語を頂点に，修飾被修飾関係を有向辺（修飾単語 → 被修飾単語）として持つグラフを作成する．コメント文のグラフ化の例を図２に示す．

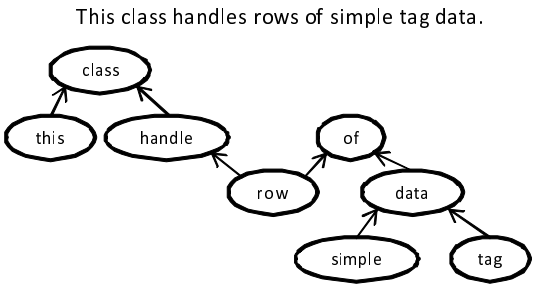

図 2　グラフ化
Fig. 2　sentence to graph

5.3.2　グラフマイニング

すべてのコメント文をグラフ化し得られたグラフ群から頻出する構造を抽出する．複数のグラフに共通して出現する部分グラフを頻出グラフマイニング手法 [5] を用いて抽出する．

あるグラフがグラフ群中のグラフに部分グラフとしてN回出現する場合，そのグラフは支持度がNであるという．グラフマイニングを行う前に最小支持度sとしてある定数を設定する．ここで，支持度N（≧s）となるようなグラフを頻出グラフという．頻出グラフマイニング手法では，グラフ群中に出現するすべての頻出グラフの抽出を行う．

また，ある単語とその単語の類義語が出現した場合，それら二つの単語は同一のものとしてマイニングを行う．類義語の判定は，英語の概念辞書であるWordNet(注1)を用いる．WordNetでは類義語を，ある概念がありその概念を含むような単語の集合という形で表現する．２つの単語がある概念に対する単語の集合にどちらも含まれている場合，その２つの単語は類義語の関係にあるとする．

5.4 説明文の生成

グラフマイニングの結果から得られた部分グラフを自然言語の文に復元することで，説明文を生成する．グラフには単語の語順は保持されておらず，正規化後の単語しか保持していない．そこで，グラフ化を行う前のコメント文を利用し自然言語文への復元を行う．まず，グラフ群から部分グラフを含むグラフを検索し，グラフとそれに対応するコメント文を取り出す．次いで，部分グラフに存在する頂点と対応する単語のみをコメント文からそのままの語順で抽出する．抽出した単語の列を説明文として出力する．

6. 手法の適用

提案手法を用いて，実際に説明文の生成を行った結果を述べる．

入力として用いたJavaソースコード集合は，ソフトウェア部品の再利用リポジトリ用に集められたオープンソースソフトウェアの集合であり，ソフトウェアプロダクト数445個，ファイル数213,159個である．説明文の生成は，入力ソースコード中の全クラス宣言の識別子中に出現した名詞すべてに対して行った．また，説明文を生成する際のグラフマイニングの最小支持度は２で行った．

6.1 結　　果

今回の結果からは，名詞の説明文としてそのまま利用できるような文を得ることはできなかった．しかし，得られた説明文の中には，名詞に関連すると思われる記述がなされている文も存在した．名詞と関連すると思われる記述の例とそうでなかった例をそれぞれ示す．

まず，得られた説明文の中で名詞と関連しそうな記述の例を図３に示す．

’exception’は例外処理に関する識別子で用いられるが，得られた説明文ではエラーの検出や制御といったように例外処理がどのようなことを行うのかについて表現している．’map’での説明文では，’map’の要素としてKeyおよびValueの２つが存在することが分かる．また，’listener’は何かしらのイベントをリッスンするものであると記述されている．そして，’handler’

(注1)：WordNet:http://wordnet.princeton.edu/

名詞	説明文
exception	・expire error specify ・control gather error
map	・estimateSet find key and value
listener	・listen message event
handle	・handler automatically further for recreate ・act call handler

図３　関連すると思われる説明文の例
Fig. 3　concern explanation

名詞	説明文
manager	・set dataTypeManager our lazy number
xml	・this is xml
comment	・@ simon.eu by parent support * and

図４　関連がないと思われる説明文の例
Fig. 4　not concern explanation

の説明文では自動的に処理を行うという記述とある動作に呼び出されるという２つの記述から，何らかの動作に起因し自動的に処理行うということが読み取れる．

次に，得られた説明文の中で名詞と関連しなさそうな記述の例を図４に示す．

’manager’の説明文では特定の識別子に関する説明が行われており，名詞に対する説明が含まれていない．次いで，’xml’では，’xml’へのの説明としてxmlが用いられおり，説明として成立していない．また，’comment’に対する説明文では，意味をなさない記述がなされている．さらに，’comment’に対する他の説明文でも，全く関連がなさそうな記述が多くなされていた．

6.2 考　　察

今回の説明文生成実験では，名詞をそのまま説明するような文を説明文として生成することはできなかった．原因としては次の２つのことが考えられる．まず１つ目が，十分な長さを持つフレーズを抽出することができなかった点がある．グラフマイニングにおいて支持度を求める際，あるグラフと同じ依存関係を持つグラフが存在する場合のみ支持度としてカウントする．そのため，似た意味のフレーズであっても，構造が少し異なると頻出グラフとして抽出できない場合がある．解決策としては，グラフ構造を比較する際に，多少の差異を許容することで，より多くのフレーズを抽出することが考えられる．２つ目としては，得られる説明文が自然言語として不完全な文であることが考えられる．その理由として，説明文を生成する際にグラフ構造に含まれる単語のみを用いるため，自然言語の文として必要な要素が欠けている場合がある．

また，図４の例で示したような，名詞と関連がないと思われる説明文が生成されてしまうことについて述べる．

特定の識別子に対する説明が生成されてしまう場合がある．識別子を含むフレーズでは，その識別子に対する説明がなされていることが多く，そのようなフレーズは説明文に利用しない

ことが考えられる．単語の説明に，その単語や似た単語が用いられ，結果的に説明を行っていないという問題は，述部に目的の単語が含むようなフレーズを抽出したために起こったと考えられる．'comment' の例のように目的の名詞に関連しない記述ばかりが説明文として生成されてしまうことがある．その原因としては，多くのコメントを収集することができず，その中に関連しそうにないコメントがいくつか含まれてしまい，そのようなコメントからばかり説明文の生成が行われてしまったためと考えられる．

7. 関 連 研 究

ソフトウェアを解析することで知識の集約を行い，それにより作業者の支援を行う研究が行われている．

Høst ら [3], [4] はメソッドへの命名支援を目的とした，メソッド名に用いられる動詞の一般的な用途を説明した辞書を，Java ソースコードを解析することで作成した．また，メソッド名の命名規則とメソッドの実装の対応を調査することで，メソッドの実装に対して不適切なメソッド名を自動的に検出する手法を提案した．Caprile ら [2] は C 言語の識別子の命名規則を定義し，関数名に使用される単語の辞書を作成した．また，その辞書を用いて関数への再命名を支援するツールを提案した．Shepherd ら [10], [11] は識別子名中の名詞と動詞をそれぞれソースコード中のオブジェクトや処理に関連付けて可視化する手法を提案した．さらに後に，識別子やコメント中に出現する動詞と目的語の組から，あるメソッドの関心事を検索するツールを作成した．Lawrie ら [6] はプログラム理解支援を目的とし，ソースコード中に出現する自然言語を解析することで，識別子に出現する略語を正式名に拡張するツールを作成した．

以上の研究に対し本研究では，Java ソースコードを解析することで，プログラム理解支援のため識別子名に用いられる名詞に対する自然言語による説明文の自動的な生成を行う．

8. まとめと今後の課題

本研究では，ソースコード中のコメントを利用することで，識別子中に出現する名詞の説明文を作成する手法を提案した

今後の課題として，説明文生成手法としての評価および説明文の品質向上の２つが考えられる．

１つ目の課題として，生成する説明文が妥当であるかの評価が必要である．評価実験として，複数の名詞それに対する生成した説明文を被験者に見せ，妥当であるかどうかを被験者が判定することで評価することを計画している．

２つ目の課題として，説明文として不適切な文が生成される場合がある．説明文作成を行う目的の単語に関する記述が含まれないコメントが多数含まれることが問題である．そのため，コメントを収集する際にフィルタリングし，そのようなコメントを減らすことが必要である．

３つ目の課題として，グラフから文へ復元した際に文の要素が足らずに，自然言語として意味が通りにくい場合がある．コメント文の構文解析時には主部や述部などの構造も取得しているので，それらを利用することで，主部がなかった場合には主部の単語を補完するなどが考えられる．

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）および文部科学省科学研究費補助金若手研究（B）（課題番号:21700031）の助成を得た．

文　　献

[1] Corbi T. A. Program understanding. In *challenge for the 1990's, IBM Syst. J.*, Vol. 28, pp. 294–306, 1989.

[2] B. Caprile and P. Tonella. Restructing program identifier names. In *Proceedings of ICSM 2000*, p. 97, 2000.

[3] E. W. Høst and B. M. Østvold. The programmer's lexicon, volume i: The verbs. In *Proceedings of SCAM 2007*, pp. 193–202, 2007.

[4] E. W. Høst and B. M. Østvold. Debuging method names. In *Proceedings of the 23rd European Conference on ECOOP 2009 – Objected-Oriented Programming*, pp. 294–317, 2009.

[5] A. Inokuchi, T. Washio, and H. Motoda. A general framework for mining frequent subgraphs from labeled graphs. In *Fundamenta Informaticae*, Vol. 66, pp. 53–82, 2005.

[6] D. Lawrie, H. Feild, and D Binkley. Extracting meaning form abbreviated identifiers. In *Proceedings of 7th IEEE International Working Conferrence on SCAM 2008*, pp. 213–222, 2008.

[7] Y. Miyao, K. Sagae, and J. Tujii. Towards framework-independent evaluation of deep linguistic parsers. In *Proceedings of the GEAF07 Workshop*, pp. 238–258, 2007.

[8] Pennington N. Comprehension strategies in programming. In *Eds. Empirical Studies of Programmers: 2nd Workshop*, pp. 100–113, 1987.

[9] Fjeldstad R.K. and Hamlen W.T. Application program maintenance study. In *Report to Our Respondents, Proceedings GUIDE 48*, 1983.

[10] D. Shepherd, Z. P. Fry, E. Hill, L. Pollock, and K. Vijay-Shanker. Using natural language program analysis to locate and understand action-oriented concerns. In *Proceedings of the 6th International Conference on AOSD 2007*, pp. 212–224, 2007.

[11] D. Shepherd, L. Pollock, and K. Vijay-Shanker. Towards suporting on-demand virtual remodularization using program graph. In *Proceedings of the 5th International Conference on AOSD 2006*, pp. 3–14, 2006.

[12] von Mayrhauser A and Vans A.M. Identification of dynamic comprehension processesduring large scale maintenance. In *IEEE Trans.Softw. Eng.*, Vol. 22, pp. 424–437, 1996.

[13] 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. 情報処理学会シンポジウム論文集, No. 3, pp. 33–34, 2008.

[14] 鹿島悠, 早瀬康裕, 真鍋雄貴, 松下誠, 井上克郎. メソッドに用いられる動詞-目的語関係を収録した辞書構築手法の提案. 研究報告「ソフトウェア工学 (SE)」, 第 2010-SE-168 巻, 2010.