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.

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

鬼塚　勇弥† 　早瀬　康裕†† 　石尾　　隆††† 　井上　克郎†††

† 大阪大学基礎工学部
†† 筑波大学システム情報系
††† 大阪大学大学院情報科学研究科
E-mail: †y-onizuk@ics.es.osaka-u.ac.jp, ††hayase@cs.tsukuba.ac.jp, †††{ishio,inoue}@ist.osaka-u.ac.jp

あらまし　プログラム理解は，ソフトウェア保守にかかるコストのうち大きな割合を占めるが，理解の過程で識別子が果たす役割は大きい．本研究では，オブジェクト指向プログラムにおけるメソッド名に着目し，メソッドを命名しようとしている開発者に対して，メソッド名の候補リストを提示することで，メソッドの命名を支援する手法を提案する．候補リストの生成には，開発者が編集中のソースコードと，既存研究で作成された動詞-目的語関係の辞書を利用する．提案手法を評価するため，被験者実験を行い，メソッドの命名が手法の有無でどのように変化するかを調査した．実験の結果，手法を用いることで，被験者の解答したメソッド名の動詞と，削除されたメソッド名の動詞とが一致する割合が高まったものの，統計的に有意な差は見られなかった．一方で，アンケート調査では，メソッド名の候補を提示するアプローチや，提示された候補リストの内容に対して好意的な意見が多いことが分かった．
キーワード　プログラム理解，動詞-目的語関係，メソッド名，命名支援

Supporting Method Naming for Java Programs Using Verb-Object Relations

Yuya ONIZUKA†, Yasuhiro HAYASE††, Takashi ISHIO†††, and Katsuro INOUE†††

† School of Engineering Science, Osaka University
†† Faculty of Engineering, Information and Systems, University of Tsukuba
††† Graduate School of Information Science and Technology, Osaka University
E-mail: †y-onizuk@ics.es.osaka-u.ac.jp, ††hayase@cs.tsukuba.ac.jp, †††{ishio,inoue}@ist.osaka-u.ac.jp

Abstract　In program comprehension, which takes the largest part of maintenance cost, identifiers have a big role in comprehension process. This paper proposes a technique to support naming a method in an object-oriented program by proposing candidates of the method name. The candidates are generated based on the dictionary of verb-object relations proposed in our past work, and the source file which a developer is editing. Effectiveness of the technique is evaluated by an experiment, which measures the difference of method naming between with/without the method. The result of the experiment cannot confirm a statistically significant difference while the average ratio of the correct answer is improved by the technique. On the other hand, the participants gave positive comment on our approach and/or the candidates of method name.
Key words　Program Comprehension, Verb-Object Relation, Method Name, Support for Naming

1. まえがき

ソースコードの読解では，識別子の名前からメソッドや変数の役割・機能を類推することが多いため，識別子に対して不適切な命名がされている場合，適切な命名が行われたソースコードに比べて理解により多くの時間がかかってしまう [1]．近年，ソフトウェアの大規模化に伴い，ソフトウェアの保守コストの増大が問題となっているが，保守作業者のソフトウェア理解の時間の増加は保守コストの増大につながる一つの要因である．

ソースコード中に出現する識別子の中で，メソッド名は，そ

れ自身の内部や，関連する識別子との間に，複雑な動詞と目的語の関係がある．そのため，開発者が類似プログラムの作成経験や開発対象のドメインの知識が不足している場合，メソッド名に対して不適切な命名をしてしまう場合がある．

そこで，本研究ではメソッドの命名に着目し，メソッド名の候補を開発者に提示することで開発者のメソッドの命名を支援する手法を提案する．

提示するメソッド名の候補を生成するにあたって，プログラム中の動詞-目的語関係に着目する．オブジェクト指向プログラムでは，あるオブジェクト A に対して操作 B を行うという動詞-目的語関係があり，既存研究には，そのような動詞-目的語関係を収録した辞書の生成手法がある [2]．この既存研究では，ソースコード中の動詞-目的語関係を表す <動詞 (V)，直接目的語 (DO)，間接目的語 (IO)> の三つ組を収録した辞書を生成している．本研究では，その三つ組を用いてメソッド名の候補を生成し，複数の基準を組み合わせることでそれらを並び替えて提示する．

提案手法を Eclipse のコード補完機能上に実装したツールを使うことで，開発者が適切な命名が行えるかどうかを確認するために行った実験について述べる．実験では，ソースコードからメソッド名などを削除した課題を用意し，複数の被験者に削ったメソッド名を解答してもらうことで，ツールの有無で正解率が変化するか比較を行った．また，提示されるメソッド名の一覧やツールの使用感に関するアンケートを行い，被験者の主観的な意見を聞いた．

以後，2. 節ではユーザにメソッド名を提示する手法を，3. 節では実装したツールの機能と使い方を，4. 節では実験とその考察を，6. 節では本研究のまとめと今後の課題に関して述べる．

2. 提案手法

本節では，返り値の型，メソッドの名前，引数の型の組 (以下，これをメソッド名と呼ぶ) の候補を開発者に提示することでメソッドの命名を支援する手法を提案する．メソッド名の候補の生成には，我々が過去に提案した動詞-目的語関係の辞書 [2] を拡張し，使用する．辞書は，メソッド名候補の生成に先立ち，作成しておく必要がある．メソッド名の候補の提示は，ソースコード中のメソッド名を記述可能な場所で，ツールを起動することによって行われる．

提案手法は，事前準備である動詞-目的語関係の辞書の作成と，メソッド名の候補リストの生成の 2 段階から成る．メソッド名の候補リストの生成は，図 1 に示すように，さらに 4 つのステップから成る．さらに細かな動作の違いとして，メソッド名候補の生成は，起動した時のソースコードの状態によって，3 通りの異なる動作をする．1 つ目はメソッドの返り値が書かれていない場合であり，2 つ目は返り値の型として void が書かれている場合であり，3 つ目は返り値の型として void 以外の型が書かれている場合である．返り値が書かれている場合には，提示されるメソッド名候補は必ずその返り値を持つものとなる．以降に記述する詳細では，簡単のため，いずれかの場合に限った説明を行う場合があるが，その他の場合においても処理に大きな違いはない．

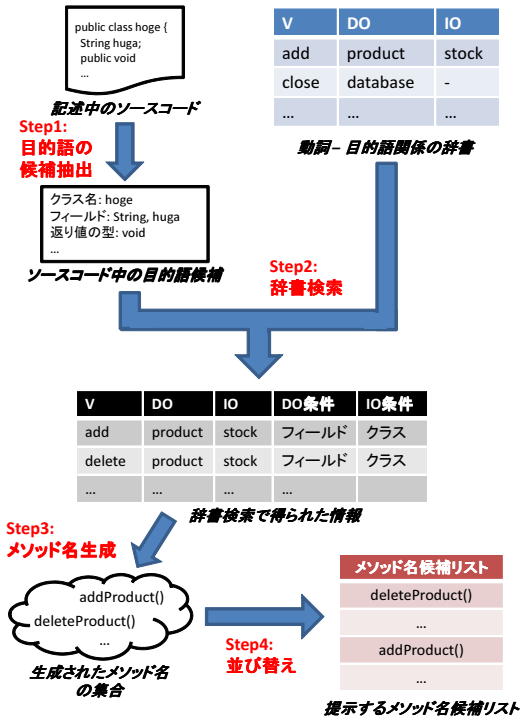

図 1　メソッド名候補の生成の流れ

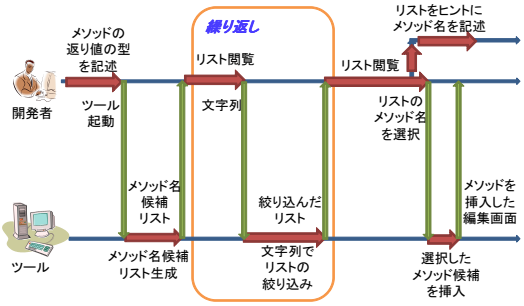

図 2　提案手法に基づいたメソッド名の入力

ツールの利用者 (ソフトウェア開発者) がメソッド名を入力するまでの手順を，図 2 に示す．ツールは起動されると，編集中のソースコードから得られる情報に基づいて，メソッド名の候補リストを開発者に返す．開発者が文字列を入力すると，リストはその文字列を含むメソッドのみに絞り込まれる．開発者はそのリストからメソッド名を選択してソースコードに挿入したり，リストをヒントにしてメソッドを記述する．

以降では，まず事前準備として辞書作成と拡張内容について説明し，その後，各ステップの詳細を説明する．

2.1 事前準備 (辞書の作成)

メソッド名候補生成の事前準備として，動詞-目的語関係の辞書を作成する．辞書は文献 [2] で我々が提案した辞書を拡張したものである．具体的には，格納する 3 つ組 <V，DO，IO> に対して，どのようなソースコードから，どのような理由で 3 つ組を取り出したのかという情報を付加する．この付加された情報は，後の段階で，利用者に提示するメソッド名候補の並び換えに利用される．

2.2 Step1. 目的語の候補抽出

Step1 では，開発者が編集しているファイルのソースコードに構文解析を行い，目的語の候補としてカーソル位置から参照可能な識別子の集合を抽出する．抽出する集合は以下の 4 つである．

- インポートクラスの名前とその親クラスの名前の集合
- 定義クラスの名前とその親クラスの名前の集合
- フィールド変数の型名と名前の集合
- 返り値の型名の集合

2.3 Step2. 辞書検索

動詞-目的語関係の辞書から Step1 で抽出した名前を検索し，検索結果情報を生成する．検索は，DO もしくは IO に対して Step1 で得た集合を割り当て，集合内の全ての名前で行う．DO と IO の両方でヒットした三つ組が出力対象となる．DO と IO の検索にどの識別子集合で行うかは，2.4 で後述するメソッド名生成パターンの検索条件に従う．

辞書検索によって生成される検索結果情報は，<検索にヒットした三つ組，その三つ組の抽出パターン，DO を検索した識別子の種類 (DO の検索条件)，IO を検索した識別子の種類 (IO の検索条件)>の組である．DO と IO の検索はどちらも前方一致で行われ，インポートクラスと定義クラスについては CamelCase のルールで単語ごとに分割し，そのそれぞれの単語でも前方一致検索を行う．検索手順は以下の通りである．

（1） DO と IO それぞれの検索条件である識別子の名前を Step1 で得た目的語候補から抽出し，それぞれを「DO で検索する名前の集合」,「IO で検索する名前の集合」とする．空なら次の検索条件へ．

（2） 抽出した集合の名前がクラス名，もしくはインポートクラス名であれば，名前それぞれを CamelCase のルールで単語に分割し集合に加える．

（3） DO で検索する名前の集合と IO で検索する名前の集合の全ての組み合わせで辞書を前方一致検索する．

（4） 辞書の項目がヒットするたびに，<ヒットした三つ組，その三つ組の抽出パターン，DO の検索条件，IO の検索条件>の組を出力する情報に加える．

2.4 Step3. メソッド名生成

Step2 で得た情報を入力とし，生成パターンに従ってその情報から<生成されたメソッド名，生成に用いた三つ組の抽出パターン>の組の集合を生成する．生成パターンとは，入力の三つ組からどのようなメソッド名を生成するか示したもので，メソッド名の生成方法は DO と IO の検索条件に割り当てられている．この生成パターンは辞書作成の抽出パターンを基に作成した．表 1 は，ユーザが書いた返り値の型が void の場合の例である．V，DO，IO は三つ組の検索結果であり，field は Step1 で抽出したフィールド名である．

表 1 返り値の型が void のときのメソッド名生成パターン

DO の検索条件	IO の検索条件	生成されるメソッド名
class	-	V()
class	-	V+DO()
class	import class	V+DO(IO)
import class	class	V(DO)
import class	class	V+DO(DO)
import class	class	V+field(DO)

2.5 Step4. 並び替え

Step3 で出力されたメソッド名を，編集中のソースコードで出現する可能性が高いものが上位に来るように並び替えるため，基準に対してそれを表す変数を定義し，各変数にパラメータチューニングによって適当な重みを与え，それぞれのメソッド名に対して重みを決定する．そして，その重みが高いメソッドを上位に配置する．出現の可能性が高いメソッド名の基準は以下の 4 つである．

（1） 多くのプログラムに出現する三つ組を使用している

（2） 三つ組を作成する時に使用した識別子の組み合わせを復元するように生成されている

（3） 汎用的な語よりそのソースコード特有の語を使用している

（4） 辞書検索のステップでソースコード中の名前と完全一致した三つ組を使用している

これらの基準から以下の 6 つの変数を用意した．

- 使用した 3 つ組がいくつのプロジェクトで使用されているか．値はプロジェクト数．(1 より)
- メソッド名の生成が三つ組抽出パターンを復元するように生成されているか．そうであれば 1，そうでなければ 0．(2 より)
- メソッド名を生成する三つ組に基本型が使われているか．使用されていなければ 2，DO と IO のどちらかが基本型であれば 1，DO と IO の両方が基本型であれば 0．(3 より)
- 辞書中に存在する，生成メソッド名と同じ動詞の三つ組数の逆数．(3 より)
- DO と IO のどちらか片方が完全一致したかどうか．したなら 1，していないなら 0．(4 より)
- DO と IO の両方が完全一致したかどうか．したなら 1，していないなら 0．(4 より)

これらの変数に重みを与え，その総和をメソッド名の並び替えに使用した．

各変数の重みを決定するパラメータチューニングでは，ソースコードで定義されているメソッドが，あるパラメータで並び替えたときにどこに表示されるかの調査を，辞書とは別のソースコード集合に対して調査し，それらの順位の逆数の和ができるだけ大きくなるようなパラメータを求めることで行った．

3. ツールの実装

本研究では前節で述べた提案手法の実装を行った．本ツールはメソッド名候補提示機能とメソッド名リストの絞り込み機能を有する．

本ツールは Eclipse のコード補完機能を拡張することで実装した．Java エディタ内で動作し，コード補完機能上にメソッド名の候補リストが表示される．

メソッド名を記述する位置にカーソルがあるときにコード補

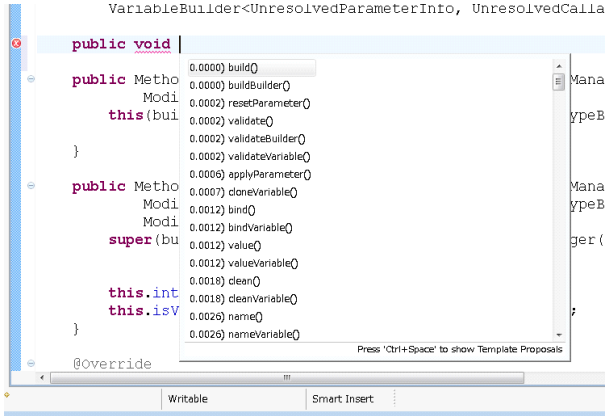

図 3　ツールにメソッド名の候補を提示させたときの様子

完を呼び出すとメソッド候補一覧が表示される．ツールは記述したいメソッドの返り値の型が書かれている場合と書かれていない場合で異なるリストを表示する．返り値の型が書かれている場合には候補のメソッド名に返り値の型が含まれない．図 3 は起動直後の画面である．リストのメソッド名は方向キーで選択し，エンターキーで挿入する．

返り値の型を記述してリストを表示した場合，文字列を入力することでリストを絞り込むことができる．文字を入力するたびに入力された文字列を含んだリストのみを返し，その後リストを選択すると入力した文字列を除いたメソッド名が挿入される．

4. 実　　験

本手法を実装したツールを使用することで，開発者がメソッドに対して適切な命名を行えるかを調査した．調査には，ソースコードからメソッド名などを削除し，複数の被験者に削除したメソッド名を推測させ，本手法を実装したツールの有無による正解率の変化を調査するという方法を用いた．また，ツールに関するアンケートも行った．本節では実験の方法，結果，及び結果からの考察を示す．

4.1 方　　法

本実験では，複数の被験者にメソッド名が記述されていないソースコードを読んでもらい，そのメソッド名を考えてもらう方法を用いた．そこで，実際に広く使われているアプリケーションのソースコードからメソッド名やそのメソッド名の直接のヒントになり得る情報を削除した課題を複数作成し，Java プログラム作成経験がある 8 人の被験者に偏りのないよう割り当てたものを解答してもらった．

また，ツールに関する被験者の主観的な意見を聞くために，実験の際には被験者に対してアンケートも行った．アンケートは各課題でリストに表示されたメソッド名について尋ねるもの，ツールを使用した感想について尋ねるものの 2 つを用意した．

4.1.1 課 題 作 成

被験者に解答してもらうメソッド名は，できるだけ豊富なものになるよう作成した．そこで，Database，GUI，Web Application，XML の 4 つのドメインのアプリケーションからそれぞれ複数作成し，各ドメインには同じ動詞のメソッド名を最大 2 つに限定した．課題作成では，被験者にメソッドの処理から名前を解答してもらうため，課題とするメソッド名以外に他のメソッド，課題とするメソッド名の動詞を含んだコメントを削除し，ローカル変数名はその変数の型名の先頭を小文字にした名前に変更した．また，ツール使用時に提示されるメソッド名候補のリストが課題に有利なものとならないように，課題作成には辞書作成とパラメータチューニングに使用したソースコード集合とは別のソースコード集合を用意した．

4.1.2 評 価 方 法

実験の評価は，ツールの有無による課題の正解率の比較とアンケートの集計結果によって行う．

正解基準は以下の 2 つとした．

- 元のメソッド名と完全一致した解答
- 元のメソッド名と動詞が一致した解答

完全一致の他に動詞一致の基準を用意したのは，メソッド名の動詞はその振る舞いを示す，メソッド名で最も重要な部分だと考えたためである．

アンケートでは被験者の主観的な意見を評価する．実験中のアンケートで，ツールありで課題を解答する際に表示された各リストが役に立ちそうかを調査し，実験終了後のアンケートで，そのツールを実験で使用してみた感想を調査する．

4.1.3 作 業 環 境

作業環境は，実際にメソッドを記述する環境で，被験者がメソッドの処理内容から推測して課題を解答するように設定した．そこで，課題の解答では，オンラインの英和・和英辞書サービスの使用を許可した．また，ツールの使い方について質問を行えるよう，作業中は筆者が部屋に待機した．一方，課題の答えを取得することを防ぐため，辞書以外のインターネットの閲覧や，課題以外のファイルを見ることは禁止した．

実験で使用した辞書の作成には SPARS [3] で使用される検索用のソースコードを用いた．これは約 400 のプログラム集合であり，これらのソースコードから約 70 万の 3 つ組を抽出した．

パラメータチューニングでは，様々なメソッド名の順位を調査するために，課題作成と同じ 4 つのドメインのアプリケーションをそれぞれ複数使用した．

4.1.4 課題の割り当て

課題は作業やツールの慣れによる正解率の偏りを防ぐため，各被験者にランダムに割り当てた．

まず，全体の作業時間や課題数を考慮して，被験者 1 人の課題数を 22 問と設定し，各問の制限時間は 3 分とした．全課題にツール有無で均等に人数を配分するため，各課題の解答人数はツールありで 2 人，ツールなしで 2 人である．

各被験者の課題をランダムに割り当てるため，どの被験者もツールありツールなしの両方の場合で 4 つのドメインそれぞれを最低 1 問解答し，同じ被験者の組み合わせの課題が同程度の数になるように各被験者に課題を割り当て，各被験者のツールありツールなしそれぞれの課題解答順をシャッフルした．被験者 8 人はランダムに割り当て，そのうち 4 人はツールありで先に解答し，残りの 4 人はツールなしで先に解答を行った．

4.1.5 実施手順

実験では被験者にツールありとツールなしのそれぞれで解答してもらった．まず，課題の解答を行う前には，実験説明，ツールの説明を行い，その後被験者には用意した練習用ソースコード上でツール使用の練習を行ってもらった．説明が終わったら，被験者に課題の解答を行ってもらった．ツールありの課題では，各課題が終わるたびに，表示されたリストに関するアンケートを回答してもらい，被験者全員がツールありツールなしの両方で全課題の解答を終了したら，最後にツールの使用感に関するアンケートに回答してもらった．

各課題の制限時間は 3 分とした．3 分たった時点で解答が思いついていればそれをすぐに記述して次の問題に，解答が思いついていなければ何も記述せずに次の問題に移ってもらった．また，3 分以内に解答が終わったら，3 分が経過するまで待機してもらった．

4.2 結　　果

本節では，課題の解答結果とアンケートの結果をそれぞれ示す．

4.2.1 課題の解答結果

ツールありでの課題とツールなしでの課題の間に有意な差があるかを調べた．ツールの有無それぞれの正解数と不正解数を表 2 に示す．この表から，2 つの基準のどちらにおいても，ツールありの方が正解率が高いということがわかる．

この結果に有意な差があるか確認するため，フィッシャーの正確確率検定でツールありの方が解答数が高いかを調べた．帰無仮説 H_0 と対立仮説 H_1 を以下に示す．

帰無仮説 H_0 ：「ツールがあるときの正解率」と「ツールがないときの正解率」に差はない．

対立仮説 H_1 ：「ツールがあるときの正解率」と「ツールがないときの正解率」に差がある．

その結果の P 値は以下のようになった．

完全一致を正解とする基準： 有意確率 $P = 0.500$

動詞一致を正解とする基準： 有意確率 $P = 0.097$

有意水準 0.05 では帰無仮説が棄却されず，ツールがあるときとツールがないときの正解率の差に有意な差は見られなかった．

4.2.2 アンケートの結果

表示されたリスト，ツールの使用感それぞれのアンケート結果について要約を示す．

a） 表示されたリストに関するアンケート

ツールで表示されたリストに対して被験者がどのように感じたかを見るため，ツールで表示されたリストに関するアンケートについて調べる．「ツールで表示されたメソッド名は、実際にそのクラスで使用されそうな名前だと思ったか」という質問に対する回答を表 3 ,「メソッド名リストに解答で用いた名前が表示されたか」という質問に対する回答を 4 に示す．

表 3 で,「そう思う」と「とてもそう思う」あわせて全回答の過半を占めることから，表示されたメソッド名の候補が編集中のソースコードに適した場合が多かったとわかる．一方，表 4 では解答に用いた名前が表示されなかったという回答が全体の 1/4 あり，生成リストに一部問題が見られる．

b） ツールの使用感に関するアンケート

被験者がツールを使用してどう思ったかを見るために，ツールの使用感に関するアンケートについて調べる．

「プログラムを書いているときにこのツールを使用したいと思ったか」という質問に対する回答を表 5 に示す．また，ツールの良い意見，悪い意見を以下に列挙する．

- 良い意見
  - 良い候補が多数あり便利だと思った
  - リストから選択しなくてもメソッドの命名の参考になる
- 悪い意見
  - 実行時間が遅い
  - 並び替えの精度がよくない

半分以上の被験者がプログラムを書いているときに本ツールを使用したいと回答しており，ツールの良かった意見なども踏まえると，ツールの使用感は悪くないものであるとわかる．一方，悪かった意見として実行時間が気になるという意見があることから，ツールの実行速度は改善の必要があることがわかる．

4.3 考　　察

実験結果で有意な差が見られなかった原因として 2 つ問題が

表 2　ツールの有無それぞれの正解数

	完全一致		動詞一致	
	正解	不正解	正解	不正解
ツールあり	2	86	16	72
ツールなし	1	87	9	79
合計	3	139	25	151

表 3　アンケート項目「ツールで表示されたメソッド名が実際にそのクラスで使用されそうな名前だと思ったか」の結果

選択肢	回答数
とてもそう思う	13
そう思う	32
どちらとも言えない	15
あまりそう思わない	22
全くそう思わない	6

表 4　アンケート項目「メソッド名リストに解答で用いた名前が表示されたか」の結果

選択肢	回答数
解答に用いた完全なメソッド名が表示された	44
解答に用いた動詞を含むメソッド名が表示された	17
解答のヒントとなるメソッド名は表示された	3
表示されなかった	24

表 5　アンケート項目「プログラムを書いているときにこのツールを使用したいと思ったか」の結果

選択肢	回答数
とてもそう思う	1
そう思う	4
どちらとも言えない	1
あまりそう思わない	1
全くそう思わない	1

考えられる．1つは実験の問題で，表2からわかるように不正解の数が非常に多いため，出題する課題が難しすぎたと考えられる．2つ目はツールの問題で，解答に用いた名前が表示されなかったという回答が全回答の1/4以上，表示されたメソッド名はそのクラスで使用されそうだと思わなかったという回答が全回答の1/4以上であり，意見にも「並び替えの精度がよくない」とあることから，候補の並び替えが適切に行えていなかったり，辞書の規模が不足しているために，被験者が適切なメソッド名を見つけられなかったと考えられる．

一方，アンケートではツールを使用したいと思う人が半数以上と高評価を得ることができた．メソッド名の候補を開発者に提示して命名支援を行うというアプローチは有効だと考える．

最後に，本実験の妥当性について検討する．課題や辞書に偏りがある場合，実験結果を実際のソフトウェア開発環境に外挿することができない．課題については，4種類の分野から注意深くメソッドを選択することで，現実に使われている多様なメソッド名を課題に採用することで解決した．辞書については，構築の際の入力に検索システムの評価用に作成したソースコード集合を用いため，有用なソフトウェアにおける命名事例をある程度バランスよく取り込んでいると期待できる．

5. 関連研究

ソースコード中の動詞や目的語を抽出していくつかの問題に適用する手法が提案されている．本節ではそれらの関連研究について述べる．

Fry らは動詞をメソッド名から，直接目的語をメソッド名中の動詞の後の語や引数，クラス名から抽出する手法を提案した[4]．Shepherd らはソースコードから取り出した動詞と直接目的語の組をグラフ化する手法を提案した[5]．また，Hill ら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と間接目的語の三つ組を抽出する方法を提案した[6]．

これらの研究に対し，本研究では，動詞と直接目的語と間接目的語の三つ組を組み合わせて，開発者にメソッド名として見える形で提示することにより，メソッド名の命名支援を行うという点が他の研究と異なっている．

6. まとめと今後の課題

本研究では，メソッドに対して適切な命名を行うことが難しいという問題に着目し，メソッド名の候補を提示することで，開発者によるメソッドの命名を支援する手法を提案した．メソッド名の生成には，ソースコードから生成した，動詞-目的語関係の辞書を利用する．本手法を用いることで，メソッドに適切な命名がなされるようになり，プログラムの可読性が向上すると期待される．

提案手法の評価として，提案手法を使用することで開発者が適切な命名を行えるようになるかを被験者実験によって確認した．結果として，ツールを用いた場合の方が適切な命名をする割合が高かったものの，ツールの有無による統計的に有意な差は見出せなかった．一方で，アンケートによって被験者からツールの使用感を聞き取ったところ，ユーザインタフェースや提示されたメソッド名の候補に対して高い評価を得た．

今後の課題として，まず今回の実験で問題があった部分を改善し，より精緻な実験を行うことで正確な評価をする必要がある．その際，アンケートで指摘された点，考察で述べたツールの問題点については改善を行う．さらに，メソッド名候補の作成に辞書以外の情報，例えばユーザが記述したテキストファイルなどを使用する，既存の悪いメソッド名を本手法で改善する，メソッド名以外の識別子の命名支援を行うと言ったツールの機能追加も検討している．

謝　辞

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

文　献

[1] D. Lawrie, C. Morrell, H. Feild, and D. Binkley, “What’s in a name? a study of identifiers,” Proceedings of the 14th IEEE International Conference on Program Comprehension(ICPC ’06), pp.3–12, 2006.

[2] Y. Hayase, Y. Kashima, Y. Manabe, and K. Inoue, “Building domain specific dictionaries of verb-object relation from source code,” Proceedings of the 15th European Conference on Software Maintenance and Reengineering(CSMR ’11), pp.93–100, 2011.

[3] “SPARS”. http://sel.ist.osaka-u.ac.jp/SPARS/index.html

[4] Z.P. Fry, D. Shepherd, E. Hill, L. Pollock, and K. Vijay-Shanker, “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.

[5] D. Shepherd, L. Pollock, and K. Vijay-Shanker, “Towards supporting on-demand virtual remodularization using program graphs,” Proceedings of the 5th International Conference on Aspect-Oriented Software Development(AOSD ’06), pp.3–14, 2006.

[6] E. Hill, L. Pollock, and K. Vijay-Shanker, “Automatically capturing source code context of nl-queries for software maintenance and reuse,” Proceedings of the 31st International Conference on Software Engineering(ICSE ’09), pp.232–242, 2009.