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

An Exploratory Analysis of Method Naming Conventions Using Association Rule Mining

柏原 由紀[*] 鬼塚 勇弥[†] 石尾 隆[‡] 早瀬 康裕[§] 山本 哲男[¶] 井上 克郎[‖]

あらまし ソースコード中の識別子がその役割を正しく表現していない場合，プログラムの理解に時間がかかることが知られている．識別子の中でも，メソッドにはその動作と対象を表すような名前を用いることが推奨されているが，一般的なガイドラインにはメソッドに対して具体的にどの単語を使って命名したほうがよいかということは明記されていない．そこで本研究では，メソッド名に使われる動詞に注目し，メソッドの命名にどのような規則があるのかを明らかにするために調査を行った．相関ルールマイニングを用いてメソッドとその名前の対応関係を自動で抽出して分析した結果，抽出したルールの中にメソッドとその名前の関係を表す規則となりうるルールがあることを確認した．また，メソッド名はメソッド内部の情報と関係が強いことを確認した．

1 はじめに

ソースコードの読解においてソースコード中の識別子がその役割を正しく表現していない場合，プログラムの理解に時間がかかることが知られている [1]．保守作業においてソースコードの読解がそのコストの大半を占めており [2]，プログラム理解に時間がかかることは保守コストの増大にもつながっている．また開発者は，ソースコード片の役割をその内部に出現する識別子から推測することがあり，保守作業の対象を特定する手掛かりの１つとしても識別子が重要であると言われている [3]．さらに，識別子の命名においてその役割を過不足なく表す命名を行うことが重要であると複数のガイドライン [4] [5] で主張されていることからも，適切な識別子をソースコード中で用いることが重要であることがわかる．

オブジェクト指向プログラミング言語の識別子には，代表的なものとしてクラス名，変数名，メソッド名が挙げられる．変数は変数が保持しているデータの意味を表す名前，メソッドはその動作と対象を表す名前，クラスはクラス内で定義されている変数やメソッドを包括するような名前をつけることがそれぞれ重要であると考えられている [4]．オブジェクト指向プログラムのメソッド名は一般に動詞や名詞などを組み合わせて命名される [4] が，一般的なガイドラインでは get，set，test といった広く知られている特定の動詞の使い方以外は，メソッドに対してどの単語を使って命名したほうがよいかということは明記されていない．メソッド名に使われる一部の動詞については，メソッドの中に記述されているソースコードの特徴が調査されている [6] が，これ以外の動詞については調査されていない．

対象とするメソッドに対してどの動詞を使うべきかという規則があれば，開発者はその規則に沿ってメソッド名を付けることができると考える．また，そのような

[*]Yuki Kashiwabara, 大阪大学大学院情報科学研究科

[†]Yuya Onizuka, 大阪大学大学院情報科学研究科

[‡]Takashi Ishio, 大阪大学大学院情報科学研究科

[§]Yasuhiro Hayase, 筑波大学大学院システム情報工学研究科

[¶]Tetsuo Yamamoto, 日本大学工学部情報工学科

[‖]Katsuro Inoue, 大阪大学大学院情報科学研究科

規則があればメソッドの内容を推測できないメソッド名に対して修正の候補を提示することに応用できると考える．

そこで本研究では，メソッド名に使われる動詞についてメソッドの命名に関する規則があるのかを明らかにするために調査を行った．多くのソースコードで頻出している関係がそのメソッドを理解しやすい規則であると仮定し，相関ルールマイニング [7] を用いて相関ルールを抽出した．メソッド名を動詞と目的語の組とみなし，メソッドを表す情報としてメソッドの定義に必要な識別子を利用することで，既存のソースファイル集合からメソッドとその名前に使われる動詞の関係を相関ルールとして抽出した．

本研究では，Java で記述されたソフトウェアを対象に 5 つのリサーチクエスチョンを設定し，分析を行った．445 個のオープンソースソフトウェアからルールを抽出し，そのルールの内容を分析するとともに，他のソフトウェアに対して適用できるかを調査した．抽出したルールの中にメソッドとその名前の関係を表す規則となりうるルールがあることを確認し，他のソフトウェアにもルールが適用できることを確認した．また，メソッド名はメソッド内部の情報と関係が強いことを確認した．

本稿における貢献は以下のとおりである．

- 既存のソースファイル集合からメソッドとその名前を対応付けるルールを抽出する手法を提案した．
- 抽出したルールからメソッド名の命名規則を分析した．
- メソッドとその名前の規則となりうるルールがあることを確認した．
- メソッド名はメソッド内部の情報と関係が強いことを確認した．

以降，2 章ではメソッドとその名前を対応付けるルールを抽出する手法について説明し，その後，3 章でリサーチクエスチョンと分析対象について述べる．4 章ではリサーチクエスチョンに基づいた分析結果を説明する．さらに 5 章では関連研究を示し，最後に 6 章でまとめと今後の課題について述べる．

2　メソッドとその名前を対応付けるルールの抽出

本研究では，相関ルールマイニングを用いて，メソッドとその名前を対応付けるルールを**命名相関ルール**として抽出し，分析を行う．

2.1　相関ルールマイニング

相関ルールマイニング [7] とは，大量のアイテム集合 (トランザクション) の集合を入力として，あるアイテムがトランザクションに出現したときに，別のアイテムもまた同じトランザクションに出現する可能性が高いという関係 (相関ルール) を抽出する手法である．相関ルールをアイテム集合 X，Y を用いて，$X \rightarrow Y$ のように表す．このとき，X を条件部，Y を帰結部といい，X，Y は入力に与えたトランザクションの部分集合となる．

抽出した相関ルールの中で相関の強さを評価する指標として，支持度と確信度という値がある．支持度は，条件部と帰結部をともに部分集合として含むトランザクションの数を表し，確信度は，条件部を部分集合として含むトランザクションのうち，帰結部もまた部分集合として含むトランザクションの割合を表す．相関ルールマイニングでは，支持度や確信度の値がある閾値以上であるときに相関があるとみなすが，この閾値は相関ルールマイニングを行うときに自由に決めることができる．

2.2　命名相関ルールの抽出

Java で記述されたソースファイル集合を入力として，相関ルールマイニングを行うことで命名相関ルールを抽出する．

相関ルールマイニング用のトランザクションはメソッド単位の解析によって取得する．解析対象のメソッド集合を M，あるメソッド m のコンテキストを $c(m)$ とす

```
import java.io.Serializable;
import java.util.LinkedList;

public class NameList
             implements Serializable {
  LinkedList<String> namelist;
  int size;

  public String findName(String n){
    if (namelist.contains(n)) {
      return n;
    }
    return null;
  }

  public void setName(Integer size) {
    this.size = size;
  }
}
```

(a) ソースコード

メソッド名の動詞：find
メソッド名の目的語：Name
クラス名：NameList
インタフェース名：Serializable
返り値の型：String
引数の型：String
引数の名前：n
フィールド名：namelist
呼び出しメソッド名：contains

(b) メソッド findName のコンテキスト

メソッド名の動詞：set
メソッド名の目的語：Name
クラス名：NameList
インタフェース名：Serializable
返り値の型：void
引数の型：Integer
引数の名前：size
フィールド名：size

(c) メソッド setName のコンテキスト

図 1　メソッドのコンテキストの例

ると，ルールマイニング用のトランザクション T は次のようなコンテキストの集合として記述できる．

$$T = \{c(m) | m \in M\} \tag{1}$$

メソッドのコンテキストとは，1 つのメソッド周辺に出現した識別子とその種別の組の集合を表すものであり，1 つのメソッドに対して一意に定まる．コンテキストに含まれる要素は，具体的には以下の通りである．

メソッド名の動詞　メソッド名に使われている動詞
メソッド名の目的語　メソッド名に使われている動詞以外の単語列
クラス名　メソッドが定義されているクラスの名前
親クラス名　メソッドが定義されているクラスの親クラスの名前
インタフェース名　メソッドが定義されているクラスのインタフェースの名前
返り値の型　メソッドの返り値の型名
引数の型　メソッドの仮引数に使われている型名
引数の名前　メソッドの仮引数に使われている名前
フィールド名　メソッドの中でアクセスしているフィールドの名前
呼び出しメソッド名　メソッドの中で呼び出しているメソッドの名前

コンテキストの例を図 1 に示す．この図の (a) のソースコードには，findName，setName という 2 つのメソッドが含まれており，findName からは (b) が，setName からは (c) がコンテキストとして得られる．

メソッド名の動詞の判定には品詞解析ができる自然言語処理ツールを利用しており，実装では OpenNLP [8] を用いた．ただし，to，new，init，calc，cleanup，setup の 6 つの単語については，品詞解析の結果にかかわらず，本手法では動詞として扱う．これらの単語は，メソッドの命名において慣習的に動詞として用いられているからである．

Java プログラムでは，言語仕様や標準ライブラリの設計によっていくつかのメソッドについては命名の規則が既に確立している．そのようなメソッドとして，本研究では以下に示すメソッドを解析対象から除外した．

main メソッドおよびコンストラクタ　言語仕様によって名前が決まっているメソッドであるため．
匿名クラス内に定義されているメソッド　ほとんどが既存ライブラリのオーバーライドであるため．
get，set の動詞が使われているメソッド　フィールドの読み書きを行うメソッドの名前として既に一般的な利用法が確立しているため．

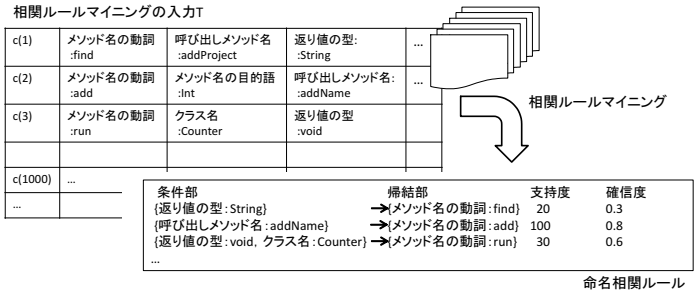

図 2　相関ルールマイニングの概要図

test の動詞が使われているメソッド　JUnit において，テストを表現するメソッドに用いることが確立されているため．

toString，**hashCode**，**equals メソッド**　いずれも Java の基底クラスである java.lang.Object に定義されたメソッドであり，オーバーライドによって使用されるメソッドであるため．

得られたトランザクション T に対して相関ルールマイニングを行う．命名相関ルールにおける支持度は，条件部と帰結部の要素がともに出現するメソッドの数となる．確信度は，条件部がコンテキストの部分集合となっているメソッドのうち，帰結部がメソッド名の動詞となっているメソッドの割合となる．相関ルールマイニングによって得た相関ルールのうち，以下に示す条件を満たすものを命名相関ルールとした．

条件 1　条件部がメソッド名の動詞・目的語以外の要素からなるもの
条件 2　帰結部にメソッド名の動詞のみが含まれているもの
条件 3　支持度が 20 以上のもの
条件 4　条件部の要素が 5 個以内のもの

条件 3 および条件 4 は，後で著者が目視で分析を行いやすいように，抽出する命名相関ルールの数を制限するために設定した．

2.3　命名相関ルールに関する用語

本研究で抽出した命名相関ルールに関する用語を次のように定義する．

命名相関ルールをメソッドに**適用する**とは，対象メソッドのコンテキストが条件を満たす命名相関ルールからメソッド名の動詞のランキングを取得することである．コンテキストが条件を満たすとは，命名相関ルールの条件部が対象メソッドのコンテキストの部分集合になっていることである．

適用の具体的な手順としてはまず，2.2 節と同様に対象メソッドのコンテキストを取得する．次に，取得した対象メソッドのコンテキストを検索条件として，あらかじめ抽出してある命名相関ルールを検索し，条件部が検索条件に用いたコンテキストの部分集合であるような命名相関ルールをすべて取得する．最後に，検索によって得た命名相関ルールを確信度によって順位づけを行い，検索によって得た命名相関ルールの帰結部に出現した動詞のランキングを取得する．複数の命名相関ルールの帰結部に同じ動詞が出現した場合，それらのルールのうち，一番確信度が高い命名相関ルールからその動詞を得たとみなす．以降の説明では，帰結部から動詞を取り出した命名相関ルールは動詞を**提供した**と表記する．

あるメソッドに対して命名相関ルールを適用したとき，対象メソッドについている動詞と同じ動詞を提供した命名相関ルールをメソッドの名前を**説明する**命名相関ルールとする．命名相関ルールを適用したとき，そのメソッドの名前を説明する命名相関ルールが得られたメソッドを**適合メソッド**とする．

対象メソッドに命名相関ルールを適用したとき，動詞を提供したすべての命名相関ルールを **fit** な命名相関ルールとする．また，fit な命名相関ルールのうち，ランキングの上位 10 位の動詞を提供した命名相関ルールを **strongfit** な命名相関ルールとし，その中でも 1 位の動詞を提供した命名相関ルールを **firstfit** な命名相関ルールとする．strongfit は fit の部分集合であり，firstfit は strongfit の部分集合である．

命名相関ルールの条件部となりうるコンテキストをメソッド内部の情報とメソッド外部の情報に分類する．それぞれ**内部要素**と**外部要素**と定義する．ここでいう外部要素はメソッドのコンテキストのうちクラス名・親クラス名・インタフェース名とし，内部要素は返り値の型・引数の型・引数の名前・フィールド名・呼び出しメソッド名とする．

3 調査

3.1 リサーチクエスチョン

本研究では，メソッドとその名前の規則を調査するにあたって，2 つの観点から 5 つのリサーチクエスチョンを設定した．まず，(1) 本手法で抽出した命名相関ルール自体の特徴を調べるという観点から，3 つのリサーチクエスチョンを設定した．次に，(2) 命名相関ルールからメソッドとその名前の規則を調査するという観点から，2 つのリサーチクエスチョンを設定した．

3.1.1 RQ1-1：命名相関ルールの抽出に用いたメソッドに使われているすべての動詞に対するルールが抽出できているか

相関ルールマイニングの入力として与えたメソッド群に対して，それらのメソッドの命名に使われているすべての動詞のルールを抽出できているかどうかを調べる．命名相関ルールの抽出に与えたメソッド群に命名相関ルールを適用し，適合メソッドの数を調べることで調査する．

3.1.2 RQ1-2：メソッドに適用するときに用いられる命名相関ルールはどのくらいあるか

命名相関ルールのうち，動詞を提供する命名相関ルールの数を調べる．具体的には，命名相関ルールの抽出に用いたメソッド群に命名相関ルールを適用し，説明するルールおよび fit，strongfit，firstfit に属する命名相関ルールの数を調査する．

3.1.3 RQ1-3：命名相関ルールは，他のソフトウェアにも適合するか

抽出した命名相関ルールの一般性を調べるために，他のソフトウェアのメソッドに命名相関ルールを適用し，適合メソッドの数を調べる．

対象としたメソッドは，命名相関ルールの抽出で解析対象となる条件に合致したメソッドである．すなわち，名前が動詞のみ，もしくは，動詞と目的語の組からなるメソッドのうち，get や set など既知の命名規則に従っているものを除いたメソッドの集合である．

3.1.4 RQ2-1：メソッド名はメソッド内部の情報に関連が強いか

メソッド名がメソッド内部の情報に基づいて命名されているのかどうかを調べるために，命名相関ルールの条件部に含まれる外部要素と内部要素の数を比較する．

3.1.5 RQ2-2：メソッドとその名前にどのような規則があるのか

メソッドとその名前にどのような規則があるのかを調べる．具体的には，命名相関ルールを第一著者が目視で 1 つずつ確認して調査を行った．

3.2 調査対象

命名相関ルールを抽出するのためのソースファイル集合として，sourceforge.net [9] および apache.org [10]，eclipse.org [11] から取得した全 445 個のソフトウェアのソースコードを用いた．これらのソフトウェアを用いて抽出した命名相関ルールの総数は，1,475,419 個である．RQ1-3 の分析対象となる他のソフトウェアとして，ArgoUML と jEdit の 2 つのオープンソースソフトウェアを用いた．

表 1　対象としたメソッドのうち元のメソッド名の動詞をルールが提示できている数

対象ソースファイル群	#LOC	#Method	#Analyzed	適合メソッド数
ルールマイニング対象	34,326,308	1,399,744	764,303	594,439 (77.8%)
ArgoUML 0.28.1	367,052	15,008	6,651	6,093 (91.6%)
jEdit 4.3.1	176,556	6,299	2,676	2,500 (93.4%)

それぞれのソースファイルの詳細は表 1 に分析結果とともに示す.

4　結果

4.1　RQ1-1：命名相関ルールの抽出に用いたメソッドに使われているすべての動詞に対するルールが抽出できているか

表 1 に，命名相関ルールの抽出に用いたメソッド群に対して，命名相関ルールがメソッドを説明できた数を示す．表の各列は，#LOC がソフトウェアの総行数，#Method が対象ソフトウェアに含まれるメソッド総数，#Analyzed が適用対象のメソッド数をそれぞれ表す.

表から，すべてのメソッドが命名相関ルールによって説明されるわけではないが，77.8%のメソッドに対して説明できていることがわかる．説明するルールが存在しないメソッドは，単純に動詞が説明できていないものもあったが，そのほとんどがメソッド名の動詞が put1 などの自動で作られたと考えられるものや，アブストラクトメソッドであるためメソッドの情報をほとんど持たず，動詞を提供する命名相関ルールの数自体が非常に少ないものであった.

4.2　RQ1-2：メソッドに適用するときに用いられる命名相関ルールはどのくらいあるか

図 3 に，命名相関ルールの抽出に用いたメソッド群に対して説明するルール，fit，strongfit，firstfit なルールにそれぞれ分類できた命名相関ルールの数を示す．横軸が分類の種類，縦軸が命名相関ルールの数を表している.

図から，抽出されている命名相関ルールの総数や入力に与えたメソッドの総数に対して，メソッドを説明する命名相関ルールの数は非常に少ないことがわかる．本手法が抽出する命名相関ルールは，実際に規則として使われるものの数が少ないことがわかる.

また，fit なルールについても命名相関ルール全体の 13.0%にとどまっている．さらに，fit なルールに属する命名相関ルールのうち 88.7%が strongfit，52.8%が firstfit にも属している．ここから，抽出した命名相関ルールにおいて，メソッドに対して適用したとき影響を与えやすい規則が偏っていることがわかる.

命名相関ルール全体で出現している動詞の種類 669 種類に対して，firstfit なルールには 401 種類 (59.9%)，strongfit な動詞については 451 種類 (67.4%) の動詞が出現している．ここから，メソッドに対して適用したとき上位に出現しやすい動詞が偏っていることがわかる.

4.3　RQ1-3：命名相関ルールは，他のソフトウェアにも適合するか

表 1 に，他のソフトウェアに対して，命名相関ルールが適合したメソッドの数を示す．表から，それぞれ ArgoUML が 91.6%，jEdit が 93.5%のメソッドが復元できていることがわかる．各ソフトウェアで適用できなかったのは 4.1 節で述べたことと同様に，単純に動詞が復元できていないものや，メソッド名の動詞が put1 などの自動で作られたと考えられるもので，命名相関ルールに対応していないもの，アブストラクトメソッドであり，メソッドの情報をほとんど持たないため適用できる命名相関ルールの数が非常に少ないものもあった.

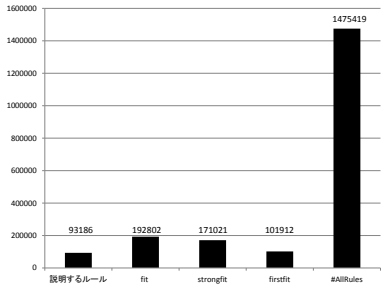

図 3　メソッド群に対して適用したときに分類されたことのある命名相関ルールの数

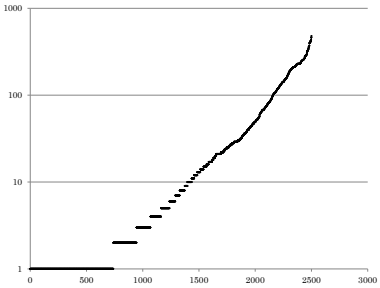

図 4　ArgoUML に対して適用したときにメソッドの名前を説明した命名相関ルールの順位

図 4，5 に，説明する命名相関ルールを確信度順に並べたときにどの位置に出現しているかを示す．それぞれの図において横軸がメソッド数，縦軸が順位を表しており，説明する命名相関ルールがその順位までに出現したメソッドの数を表している．また，縦軸は対数軸で表している．図から，説明した命名相関ルールのうち ArgoUML で 62.0%，jEdit で 57.3%がランキングの 10 位以内に出現している．確信度が高い命名相関ルールが他のソフトウェアでも多くのメソッドの名前を説明しており，異なるソフトウェアに対しても命名に共通性があると考えられる．ここから，命名相関ルールは抽出に用いたメソッド群以外にも適合するといえる．

4.4　RQ2-1：メソッド名はメソッド内部の情報に関連が強いか

図 6 に，命名相関ルールの条件部における内部要素と外部要素の数の分布を示す．横軸が内部要素または外部要素の数を表している．棒グラフが，条件部に横軸に対応する数の外部要素または内部要素を持つ命名相関ルールの数を表し，その縦軸は図の左側に示している．折れ線グラフが内部要素または外部要素の数に対する命名相関ルールの累積度数を表し，その縦軸は図の右側に示している．

図から，メソッド名はメソッド内部の情報に関連が強いことがわかる．メソッド外部の情報を含む命名相関ルールが全体の 25.6%で，まったく含まない命名相関ルールが全体の 74.3%と一番多いのに対し，メソッド内部の情報は 99.2%の命名相関ルールに使われており，4 つ以上含む命名相関ルールが全体の 51.5%と一番多い．

内部要素が関連している理由としては，外部要素の数より内部要素の数の方が多い点があげられる．また，メソッド名はその処理内容を表すようにつけることが重要だとされているため，実際の動作を記述しているメソッド内の情報と関係が強いと考えられる．

外部要素が関連しているルールは，インタフェースによるメソッド実装や継承によるオーバーライドによって名前を変更できないメソッドに対するものが多かった．

4.5　RQ2-2：メソッドとその名前にどのような規則があるのか

抽出した命名相関ルールからどのような規則があるのかを分析した．特に相関ルールマイニングの入力に与えたメソッド群を説明する命名相関ルールについて分析を行った．その結果，4 つの規則を見つけた．(1) メソッド内部で呼び出しているメソッドの動詞と関係しているという規則，(2) メソッド内部で使われるデータの種類と関係しているという規則，(3) メソッド外部の情報に関係しているという規則，(4)Java の典型的な処理に関係しているという規則である．規則に対応するいくつかのルールを例として表 2 に示す．表において cmn は呼び出しメソッド名，rt は返り値の型，argn は引数の名前，argt は引数の型，cn はクラス名を表す．

表における R1，R2，R3 は，内部で呼び出したメソッドとその名前の動詞が同じである規則の例である．R1 と R2 の命名相関ルールは create の動詞を提供して

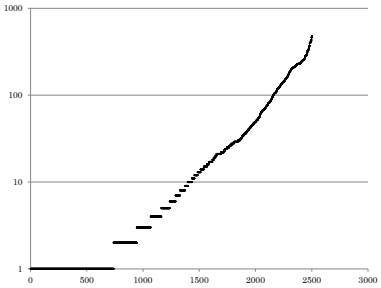

図 5　jEdit に対して適用したときにメソッドの名前を説明した命名相関ルールの順位

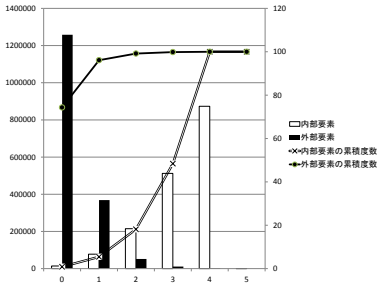

図 6　命名相関ルールの内部要素・外部要素の要素数とその累積度数分布

表 2　メソッドを説明する命名相関ルールの例

ID	条件部	帰結部	確信度	支持度
R1	cmn:createInstance，cmn:getCimObjectPath	create	1	525
R2	rt:String，cmn:createFile	create	0.492069	31
R3	cmn:containsValue，rt:boolean，argt:Object	contains	0.836957	77
R4	cmn:getCimInstance，argn:dataInstance argt:client，cmn:checkDifferencesAfterCreate	create	1	525
R5	cn:ASTFlatter	visit	1	30
R6	cmn:hasNext，cmn:iterator，argt:String，argn:paren	find	0.766667	23
R7	rT:boolean	applies	0.000529	59

いるが，呼び出しメソッド名に create の動詞が使われていることがわかる．R3 は，contains の動詞についての同様の例である．

また R4 は，メソッド名がメソッド内部で用いるデータの種類と関係していると考えられる規則の例である．create の動詞を提供している R1 と R4 の命名相関ルールは Instance という単語が共通している．ここから，Instance という単語と create の動詞に関係があると考えられる．

R5 は，RQ2-1 で述べた外部要素がメソッド名に関係している例である．AST-Flatter というクラスが，ASTParser というクラスを継承して多数の visit メソッドをオーバーライドしていると考えられる．

R6 は，Java の典型的な処理と名前が対応している例である．find という動詞が iterator メソッドと hasNext メソッドを用いて順にリストを検索してある要素を探すメソッドを表す動詞であると考えらえる．またこの命名相関ルールは，Høst ら [12] が紹介した find の動詞を使うメソッドの特徴と類似している．

一方で，動詞によっては直感的な規則が存在しないものもあった．R7 の命名相関ルールである．applies という動詞について抽出できた命名相関ルールは 1 個だけであるが apply という動詞については 3,194 個抽出されたことから，applies があまり命名に使われていない例外的な動詞だと考えられる．

結果として，抽出した命名相関ルールには納得できそうな規則が含まれていることを確認できた．これらの規則を用いて，メソッドの内容に対応したメソッドの命名や，名前の変更リファクタリングの支援を行える可能性がある．

4.6　妥当性への脅威

命名相関ルールの抽出には，特定のソースファイル集合を用いた．ソースファイル集合は，多様性を高めるように選択したが，意図しない偏りがある可能性がある．そのため，命名相関ルールの抽出に用いるソースファイル集合を変更した場合，分

析結果が変わるかもしれない．

メソッドとその名前に規則があるかを調査するために，相関ルールマイニングの結果を分析した．メソッドとその名前の間の規則は，本研究で得た結果と他のアプローチで得た結果に差が出るかもしれない．

メソッド名の動詞を判定するために OpenNLP を用いた．自動で判定しているため，本来動詞ではない単語を動詞として扱っている可能性がある．しかしそのような単語に対しても命名相関ルールが抽出されており，そのようなルールの数も少ないため，結果に大きな影響はないと考えられる．

命名相関ルールの順位付けに確信度の値を用いた．順位付けに確信度以外の値を利用したり，他の値と組み合わせたりすることで結果が変わるかもしれない．

命名相関ルールの適用対象ソフトウェアとして，ArgoUML と jEdit を選択した．2 つのソフトウェアはどちらも著名なオープンソースソフトウェアであり，他の研究でも評価に用いられているため利用した．しかし，そのどちらも GUI アプリケーションであり，対象としたドメインが偏っている可能性がある．また，対象が 2 つのソフトウェアであるため，より多くのソフトウェアを対象とすることで結果が変わる可能性がある．

命名相関ルールのどれだけが実際に規則であるかどうかの定量的な分析は行っていない．これは，本研究では著者が目視で命名相関ルールの分析を行ったからである．規則となっていると考えられる命名相関ルールは著者の経験から判断した．

5　関連研究

Høst らは，メソッドの命名に使われる動詞とメソッドの動作の関係を調査している [6]．この研究では，著者らが重要であると考えた 40 種類の動詞を対象としていたが，本研究ではルールが抽出できた 669 種類の動詞について調査した．また，Høst らは Java のバイトコードから制御構造を抽出して関係を調査しているのに対し，本研究ではソースコードから識別子を抽出して関係を調査している．バイトコードの解析では，コンパイラが生成したメソッド呼び出しなどが対象に含まれるが，ソースコードの解析では開発者が記述した内容をそのまま解析できる．

Høst らは，メソッド名の動詞に着目して，不適切な動詞がついているメソッド名に対して，適切な動詞を理由とともに提示して開発者のメソッドの命名を支援する手法を提案している [12]．この手法は，あらかじめメソッド名の動詞とメソッド本体を対応付けたルールを作成しておき，既についているメソッド名の動詞とメソッド本体がこのルールを満たしていない場合，不適切である理由とともにメソッド本体に対応した動詞を提示する．この手法は不適切な命名を見つける手法であり，作られているルールを満たすことは命名が不適切であることを示す．これに対して本研究の分析手法では，メソッドに対して適切な動詞が何かを示すルールを抽出する．

鹿島らは，オブジェクト指向プログラムのソースコードに出現するメソッド名から動詞-目的語関係を抽出して辞書を作成し [13]，鬼塚らはその辞書を用いて，新規に記述するメソッドの命名を支援する手法を提案した [14]．本研究では，この研究からメソッド名を周辺の識別子から取得するというアイデアを得た．また，本研究で抽出した命名相関ルールは，鬼塚らの手法と組み合わせることで提案するメソッド名の精度を高めることに利用できるのではないかと考える．

鬼塚らは，メソッド名やメソッドの定義位置を利用してメソッド本体を補完する手法を提案した [15]．この研究では，メソッド外部の識別子とメソッド名を利用して，メソッドの内容の候補を導き出す試みが行われている．

6　まとめと今後の課題

本研究では，既存のソースファイル集合から相関ルールマイニングにより抽出した命名相関ルールを用いて，メソッドとその名前に使われる動詞の間に規則がある

のかどうかを調査した．その結果，4 つの規則が存在していることを確認した．(1) メソッド内部で呼び出しているメソッドの動詞と関係しているという規則，(2) メソッド内部で使われるデータの種類と関係しているという規則，(3) メソッド外部の情報に関係しているという規則，(4)Java の典型的な処理に関係しているという規則である．メソッド名はメソッド内部の情報に関係が強いことも示した．また，使われていない命名相関ルールも多いが，命名相関ルールの抽出に使っていない 2 つのソフトウェアに対して適用して動詞を説明することができた．

今後の課題としてはまず，使われていない命名相関ルールのフィルタリングが挙げられる．今回，相関ルールマイニングを行うときに，確信度の閾値を定めなかった．そのため，確信度の閾値を変更することで使われていない命名相関ルールを削除できる可能性がある．また，今回用いなかったリフト値を用いて解決できる可能性がある．リフト値は相関ルールマイニングにおいて，ある相関ルールの支持度とその相関ルールの条件部を空にしたときの支持度との比率を表す値である．

メソッド名の目的語についても動詞と同様にメソッドの内容と関係があるかどうか，また，今回定義したコンテキスト以外のソースコード中の要素とメソッド名との関係についても分析する必要がある．

そして，本研究で示した命名相関ルールから得た規則をメソッドの名前変更リファクタリングの支援やメソッドの命名の一貫性の検査に利用することが考えられる．

謝辞

本研究は，科研費 (23680001，25220003，25730036) の助成を受けたものである．

参考文献

[1] Dawn Lawrie, Christopher Morrell, Henry Feild, and David Binkley. What's in a name? a study of identifiers. In *Proceedings of the 14th IEEE International Conference on Program Comprehension*, pp. 3–12, 2006.

[2] Gail C. Murphy, Mik Kersten, Martin P. Robillard, and Davor Čubranič. The emergent structure of development tasks. In *Proceedings of the 19th European Conference on Object-Oriented Programming*, pp. 33–48, 2005.

[3] Andrea De Lucia, Massimiliano Di Penta, Rocco Oliveto, Annibale Panichella, and Sebastiano Panichella. Using IR methods for labeling source code artifacts: Is it worthwhile? In *Proceedings of the 20th IEEE International Conference on Program Comprehension*, pp. 193–202, 2012.

[4] Steve McConnell. *Code Complete, Second Edition*. Microsoft Press, Redmond, WA, USA, 2004.

[5] Sun Microsystems, Inc. *Code Conventions for the Java Programming Language*, 1997.

[6] Einar W. Høst and Bjarte M. Østvold. The Programmer's Lexicon, Volume I: The Verbs. In *Proceedings of the 7th IEEE International Working Conference on Source Code Analysis and Manipulation*, pp. 193–202, 2007.

[7] Rakesh Agrawal, Tomasz Imielinski, and Arun Swami. Mining association rules between sets of items in large databases. In *Proceedings of the 1993 ACM SIGMOD International Conference on Management of Data*, pp. 207–216, 1993.

[8] OpenNLP. `http://opennlp.sourceforge.net/`.

[9] SourceForge. `http://sourceforge.net/`.

[10] Apache Software. `http://www.apache.org/`.

[11] Eclipse. `http://www.eclipse.org/`.

[12] Einar W. Høst and Bjarte M. Østvold. Debugging Method Names. In *Proceedings of the 23rd European Conference on Object-Oriented Programming*, pp. 294–317, 2009.

[13] 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. 情報処理学会研究報告, Vol. 2010-SE-168, No. 12, pp. 1–8, 2010.

[14] 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.

[15] Yuya Onizuka, Yasuhiro Hayase, Tetsuo Yamamoto, Yuki Kashiwabara, Takashi Ishio, and Katsuro Inoue. Towards generating templates of method body based on method name and related identifiers. In *Proceedings of the 8th International Workshop on Advanced Modularization Techniques*, pp. 13–14, 2013.