Vol.2012-SE-176 No.6
Vol.2012-EMB-25 No.6
2012/5/21

ソフトウェア部品検索のための API を利用した類似部品検出方法の提案

高見愛[†]　井上勝行[†]　北村操代[†]

本稿では、ソフトウェア部品が提供する公開 API の型や識別子の類似性を利用して部品間の類似度を算出する方法を提案する。これにより、ユーザ所望の機能を表すキーワードを与えたとき、そのキーワードに該当する部品を検出するだけでなく、その部品に類似する部品も候補として提示できるようになる。さらに、自社製 C++ライブラリを対象として、算出した類似度の妥当性を検証する。その結果、クラスの継承関係が小さい部品間の類似性を検出できるなど、提案手法によって、類似部品を抽出できることが示唆された。一方、技術者による類似部品の判定結果との差異も見られたため、今後は他のメトリクスと併用も検討していく。

Method of Extracting Similar Software Component using API for Retrieving Software Component

AI TAKAMI[†]　KATSUYUKI INOUE[†]
MISAYO KITAMURA[†]

This paper proposes a method of calculating the degree of similarity between the software components using similarity of their public APIs in type and identifier. Our aim is to extract functional similar components in addition to components which are associated with keywords given by the user. And experimentation is performed for evaluating the degree of similarity calculated by our approach. The subject of the experimentation is C++ class libraries developed by our company. The result suggests that our method can extract similar components which have a weak inheritance relationship. On the other hand, the result shows that similar components judged by skilled engineers are sometimes different from ones extracted by our method. It suggests a need for applying other software metrics to our calculating method.

1. はじめに

企業におけるソフトウェア開発では、過去に製作したソースコードなどの成果物を流用して効率化を図ることが求められているが、蓄積された成果物の中から、システム設計者が所望のものを探すことは容易ではない。成果物を再利用可能な単位でソフトウェア部品として管理し、所望の機能を表現する語句（以降、キーワードと呼ぶ）を入力して検索する方法が考えられる。しかし、適切なキーワードを選定することや、部品の有する機能全てを把握してキーワードを付加することは難しいため、キーワードの表記揺れや不足を考慮しなければ所望の結果が得られない。特に、企業においては、納入先の顧客が異なると用語も異なることが多く、表記揺れの影響を受けやすい。

適切なキーワードの付加は困難であるため、キーワードによらない検索方法[1][2][3]や、キーワードによる検索結果を補足する方法[4][5]が研究されている。例えば、前者には、形式手法に基づいて部品の検索を行なう方法[1]や、インタフェースのみを定義した部品のプロトタイプを入力として、置き換え可能なプログラムを検索する方法[2]がある。また、例えば、後者には、キーワードによる検索結果に合わせて利用関係のある部品やコピーされた可能性のある部品を提示する方法[4]、部品が対象とする業務の類似性に基づき部品をマッピングして表示する[5]方法がある。

キーワードによらない検索方法は、人の知識だけでは検索できない部品も検索でき有用であるが、所望の部品に対してコーディングレベルでの理解が必要となる。そのため、ソフトウェアの詳細な仕様が決まっていない状態での検索は難しい。顧客からの要求仕様が決まった段階で、システム設計者が流用できる部品を探すためには、詳細な仕様が決定していなくても流用できることが望ましい。

一方、検索結果を補足する方法では、部品の利用関係がある場合[4]や業務が類似している場合[5]など、部品間に明示的な関連性が与えられている場合に有効なため、部品に直接または間接的に利用関係がない場合や、顧客の分野が全く異なる場合には有効ではない。

本稿では、ソフトウェア部品が提供する公開 API の類似性を利用して、機能の類似するソフトウェア部品を抽出する方法を提案する。これにより、キーワードによる検索に加え、機能の類似するソフトウェア部品も候補として提示できるので、ソフトウェア部品に対応づけられているキーワードに表記揺れや不足があり、さらに部品間の関連性が明示的に与えられていない場合でも、表記揺れや不足による悪影響を軽減した検索が可能になる。

また、提案手法をソフトウェア部品間類似度算出ツール Collate++として実現し、自社製の C++クラスライブラリを対象として、類似度の妥当性を検証する。検証では類似す

[†] 三菱電機株式会社 先端技術総合研究所
Advanced Technology R&D Center, Mitsubishi Electric Corporation.

る部品の数が多い部品と少ない部品の特徴が類似性を反映しているか、別のライブラリにおいて同様の機能をもつ部品が類似しているとされるか、ライブラリを熟知した技術者による類似性の感覚との違いについて調査する。

本稿は以下のように構成される。2 章では、ソフトウェア部品間の類似度の算出方法について説明する。3 章では、ソフトウェア部品類似度算出ツール Collate++について説明する。さらに、4 章では、自社製ライブラリを対象として、類似部品を検出した結果を考察する。最後に、5 章で本稿をまとめる。

2. ソフトウェア部品間の類似度

本章では、ソフトウェア部品間の類似度を算出する方法について述べる。

2.1 既存の類似度算出手法

ソフトウェアの類似度を算出する方法には、ソースコード中の最長一致文字列を抽出する方法や、コードクローンを利用するなどしてプログラムの構造を比較する方法[6]、ソフトウェア部品の外部仕様を利用する方法[1][3]がある。ソースコード中の最長一致文字列を抽出する方法やプログラムの構造に着目する方法は、部品の実装方法に強く依存するため、部品のコピーを抽出するのに適しているが、異なる開発者が作成した部品を抽出できない。

外部仕様を利用する方法では、部品が外部に提供するインタフェースや外部に依存するインタフェースを利用するため、部品が外部に提供する機能における類似性を検出できる可能性がある。ただし、外部仕様を利用する既存の方法では、これから作ろうとする部品のプロトタイプを入力として、置き換え可能な部品や変更を加えれば適用可能な部品を探すことを目的としているため、本稿で用いるには厳密すぎて、機能の一部のみが似ている部品を検出できない。

2.2 既存の類似度算出手法

本稿では、類似する部品の抽出に、部品の外部仕様のうち部品が公開する API に着目し、API 中の型と識別子の類似度に基づいて、ソフトウェア部品の類似度を算出する方法を提案する。API が異なっていても、関数や変数、引数の型、識別子のいずれかが同一または類似する場合に API の果たす機能が類似すると見なして、類似度の得点を加算し、最終的な得点がしきい値を越えると API が類似すると判定する。さらに、ソフトウェア部品に対して、その部品の公開 API に類似する API を多くもつ部品を類似部品と判定する。

既存の研究でも、Michael らが、クラス名やクラスのメンバ関数に現れる語句を利用して部品間の類似度を算出している[7]が、クラスと関数、関数と変数などの異なる種類間の比較は行なっていない。本提案手法では、関数や変数の区別をせずに比較を行ない、同一または類似する型や識別子が用いられていれば機能が類似すると見なす。例えば、変数 “int gradation_color”と、関数“void FillGradation()”は共に、gradation という単語が含まれることから、共にグラデーションに関する機能を果たすと考え、類似度の得点を加算する。

2.2.1 API の類似性

API の類似性は、二つの観点により判定する。一つ目は API の型である。型は、関数の戻り値の型、または変数の型に加え、引数の型を含む。これらの型が同一もしくは互換である場合、API の入出力となるデータが同じもしくは似ていると見なして、API は似た機能を有すると判定する。

二つ目の観点は、API の識別子である。識別子は、関数の名前と変数の名前、および引数の名前を含む。識別子が同一もしくは互換であれば、API は似た機能を有すると見なす。多人数がコーディングに携わる場合、関数や変数の命名方法は、その役割をよく表すような名称をつけることをコーディング規約で促されることが多いため[8][9]、API 中の識別子が類似していれば、API は似た機能を有すると仮定する。また、識別子は複数の単語が組み合わさった場合を考慮し、要素に分割して比較を行なう。例えば、関数名“OpenFile”であれば、“Open”と“File”に分割する。識別子は類語辞書を入力として与え、類語と判定された場合に類似していると見なし、文字単位での比較は行なわない。

API の類似度は、正規化のため、全く同じ機能を持たない API 間の類似度は 0、同一の API 間の類似度は 1 とし、類似度は 0 から 1 までの値をとることとする。API の類似度は、型に関する類似度の得点と、識別子に関する類似度の得点を足し合わせ、取り得る得点の最大値で割ったものとする。API の類似度がしきい値を超えたとき、API は類似していると見なす。この閾値を API 間類似度判定しきい値と呼ぶ。

2.2.2 ソフトウェア部品の類似性

ソフトウェア部品の類似性は、類似する API の割合によって判定する。具体的には、部品 P の部品 Q に対する類似度は、部品 P の公開 API のうち、部品 Q に類似する API が存在するものの割合とする。この類似度が高い場合、部品 P は部品 Q に類似していると見なす。このとき、部品 Q は部品 P の類似部品と呼ぶ。類似部品の判定にはしきい値を利用する。このしきい値を以降、部品間類似度判定しきい値と呼ぶ。しきい値は経験的に定めるものとする。

API の類似度と同様に、正規化のため、全く同じ機能を持たない部品間の類似度は 0、同一の部品間の類似度は 1 とし、類似度は 0 から 1 までの値をとることとする。

2.3 部品の類似度の算出手順

ここでは、部品 P から部品 Q への類似度を算出する手順を示す。

手順a1) 部品 P と部品 Q より公開 API の集合を取得する。

手順a2) 部品 P の公開 API と部品 Q の公開 API を、全ての組合せにおける API 間類似度を算出する。API の類似度の算出手順は後述する。

手順a3) 部品 P のある 1 つの公開 API において、部品 Q の公開 API に対する類似度が、予め定めた API 間類似度判定しきい値を超えた場合に部品 P の部品 Q への類似度の得点を 1 点加算する。

手順a4) 手順 a3 を部品 P の公開 API 全てに対して行なった後の得点を部品 P の API 数で割る。得られた値を部品 P から部品 Q への類似度とする。

次に、具体的に API α と API β の類似度を算出する手順を以下に示す。

手順b1) API α と API β より型と識別子を抽出する。

手順b2) API α と API β の取り得る得点の最大値を算出する。API α が取り得る得点の最大値を API α の持ち点とする。

手順b3) 抽出した双方の型と識別子を順に比較し、類似度の得点を加算する。

手順b4) 手順 b3 を双方の全ての型と識別子に適用した後の得点を、手順 b2 で求めた持ち点の最大値で割る。得られた値を API α と API β の類似度とする。

3. ソフトウェア部品類似度算出ツール Collate++

本章では、2 章で提案した類似度の算出方法に基づき、ソフトウェア部品の類似度を算出するツール Collate++を実装した。なお、Collate++は C++言語で開発されたソフトウェア部品を対象とする。

3.1 部品類似度算出処理の実現

Collate++はヘッダファイルで宣言される API のうち、次の関数および変数を抽出する。変数はグローバル変数と、public および protected 宣言されたデータメンバを抽出する。関数は、グローバル関数と、public および protected 宣言されたメンバ関数を抽出する。

型の類似度は、型の互換性に基づき、表 1 の 6 種類の中から選択する。その得点との対応を表 2 に示す。ただし、表 1 ではクラス Derived はクラス Base を直接あるいは間接的に継承するクラスとする。

int や char などの、組込みの型の類似度は、組込み型類似度ファイルを利用する。その他の型の類似度は、型がソースコード中で宣言されたクラスである場合、クラスの継承関係を元に類似度を求める。二つのクラスが直接または間接に継承関係にある場合、表 1 の A, B, C, F のいずれかに分類する。継承関係と型の類似度の対応は、継承クラス類似度ファイルを利用する。組込みの型でなく、かつ継承関係にない型は識別子と見なして類似度を算出する。なお、組込み型類似度ファイルと継承クラス類似度ファイルは、ユーザが作成したものを利用する。

表 1　型の類似度の具体例

カテゴリ	意味	具体例	
		種類	説明
A	型 to と型 from は同一である。または、型 to は型 from と互換であり、値域は型 to の方が広い。	組込み	型 to の値域が型 from よりも広い。 例) 型 from が char、型 to が int
		クラス	型 from がクラスのポインタ型、型 to がそのスーパークラスのポインタ型である。 例) 型 from が Derived*、型 to が Base*
B	型 to は型 from と一部互換であり、型 from の値域で型 to の値域に含まれない範囲がある。	組込み	型 from の値域が型 to よりも広い。 例) 型 from が int、型 to が char
		クラス	型 from がクラス、型 to がそのスーパークラスである。 例) 型 from が Derived、型 to が Base
C	A と B 以外で、型 to は型 from と似ているが、型 from から型 to へ安全にキャストできないもの。	組込み	1) unsigned の有無を除けば、A または B と同一。 2) const 修飾子の有無を除けば、A または B と同一。 3) 型 from と型 to をどちらもポインタ型にすれば、A または B と同一。
		クラス	・型 from と型 to を入れ替えれば A または B と同一。 例) 型 from が Base*、型 to が Derived* ・組込み 2）と同様 ・組込み 3）と同様
D	A～C 以外で、型 from と型 to は歴史的経緯から似ているものとする。	組込み	過去に型 from の代用として型 to が用いられたことがある。 例) 型 from が void*、型 to が char*
		クラス	該当しない
E	A～D 以外で型 from を型 to へキャストしても安全かもしれない。	組込み	慣例として型 from を型 to へキャストして用いることがある。 例) from が void*、to が int*
		クラス	該当しない
F	型 to と型 from は全く似ていない。	組込み	A～E のいずれにも当てはまらないもの
		クラス	A～E のいずれにも当てはまらないもの

表 2　型の類似度の得点

カテゴリ	得点
A	5（満点）
B	4
C	3
D	2
E	1
F	0

識別子の類似度は、二つの識別子が類語関係にあるかどうかで判定する。識別子の類似度の得点は、表 3 に示す。二つの識別子が類語関係にあるかは、類語辞書ファイルで類語として記載されているかどうかで判断する。図 1 に類語辞書ファイルの例を示す。この図では、”string”と同じ行に表れる識別子の”str”、”wstring”、”wstr”および”mem”と類語関係にあることを示す。ただし、”mem”が二ヶ所に現れるが、memory と string の間には類語の関係はないものとする。なお、類語辞書ファイルとして 21791 行からなるファイルを用意した。なお、マクロを利用して宣言される API も抽出するために、ヘッダファイルは事前にプリプロセッサにかけたものを利用する。ただし、include 命令が展開された範囲からは API は抽出しない。

表 3　識別子の類似度の得点

類似度	得点
同一	2（満点）
類似	1
類似していない	0

```
compare      cmp
destination  dest
memory       mem
source       src
string       str     wstring   wstr   mem
```

図 1　類語辞書の例

3.2 API 類似度の算出例

関数 strcpy の関数 memcpy に対する類似度を算出した具体例を示す。なお、strcpy の関数プロトタイプを char* strcpy(char *dest, const char *src)、memcpy の関数プロトタイプを void* memcpy(void *dest, const void *src, size_t n)とする。

① 持ち点の算出
プロトタイプから型と識別子を抽出し、それぞれの関数が取り得る得点の最大値を求める。表 4 に関数 strcpy の持ち点を、表 5 に関数 memcpy の持ち点を示す。

② 型の類似度と識別子の類似度の算出
型と識別子をそれぞれ順に比較し、類似度の得点を算出する。最後に得点を全て加算する。

③ 表 6 に得られた得点を示す。

④ API の類似度の算出
②で得られた得点(22)を、①で得られた持ち点の最大値(30)で割った値が strcpy の memcpy に対する類似度となる。この例では、strcpy と memcpy の類似度は 11/15 となる。

表 4　関数 strcpy の持ち点

要素	種類	満点
char *	型	5
str	識別子	2
cpy	識別子	2
char*	型	5
dest	識別子	2
const char*	型	5
src	識別子	2
合計（持ち点）		23

表 5　関数 memcpy の持ち点

要素	種類	満点
void*	型	5
mem	識別子	2
cpy	識別子	2
void*	型	5
dest	識別子	2
const void*	型	5
src	識別子	2
unsigned int	型	5
n	識別子	2
合計（持ち点）		30

表 6　strcpy と memcpy の類似度の得点

要素	要素	類似性	得点
char*	void*	A	5
str	mem	類似	1
cpy	cpy	同一	2
char*	void*	A	5
dest	dest	同一	2
const char*	const void*	A	5
src	src	同一	2
	size_t	比較なし	0
	n	比較なし	0
		合計	22

4. 既存ライブラリを対象とした類似度の算出

自社製の C++クラスライブラリを対象として、2 章で述べた算出方法で算出した類似度を評価する。

4.1 対象としたライブラリ

対象とするクラスライブラリ（以降、評価ライブラリと呼ぶ）は以下の特徴をもつ。

- GUI 機能をもつフレームワークであり、画面に表示するボタンやフィールドなどのクラスや、表示すべきデータを格納するモデル用のクラスなどから構成される。
- 10 年以上前より多数の製品に適用された実績がある。
- 評価ライブラリの規模は表 7 に示すとおりである。

Vol.2012-SE-176 No.6
Vol.2012-EMB-25 No.6
2012/5/21

表 7　評価対象の C++ライブラリ

ソースファイル	200 件
うちヘッダファイル	102 件
行数（コメント行を除く）	83 k 行

4.2 類似度の算出

評価ライブラリのソフトウェア部品間の類似度を計測する。計測結果について、以下の 3 つの観点で調査を行なう。

(1) 抽出された類似部品の数

抽出された類似部品の数と、ソフトウェア部品の特徴について考察する。以降では、部品 P が部品 Q の類似部品である場合に、部品 Q は部品 P の被類似部品と呼ぶことにする。

(2) 別ライブラリを利用した類似部品の抽出

評価ライブラリとは異なるライブラリ（以降、別ライブラリと呼ぶ）を用いて、評価ライブラリと同様の機能をもつ別ライブラリの部品を類似部品として抽出できるかを調査する。本稿では、画面編集用ライブラリを別ライブラリとした。入力フィールドを表す部品群と、ボタンを表す部品群を利用する。

(3) 熟練技術者により選出された類似部品との比較

評価ライブラリを熟知している技術者 2 名に各ヘッダファイルに対する類似部品を選出してもらい、選出された部品と、Collate++によって抽出された類似部品を比較する。なお、熟練技術者 2 名は共に評価ライブラリを利用した開発に 10 年以上携わった経験がある。

なお、ソフトウェア部品の単位は、1 ヘッダファイルと、そこで宣言されている API が定義されているソースコードをまとめたものとする。

また、事前調査で類似部品として抽出される部品数に基づいて検討を行ない、API 間類似度判定しきい値と部品間類似度判定しきい値は共に 0.5 とする。

4.3 算出結果

ソフトウェア部品間の類似度を算出した結果を以下に示す。

4.3.1 抽出された類似部品の数

図 2 に、各ソフトウェア部品の類似部品数の度数分布を示す。次に、図 3 に各ソフトウェア部品の被類似部品数の度数分布を示す。類似部品は、最も多いもので 31 個の部品が抽出された。これに対し、被類似部品が最も多い部品は 18 個の部品が抽出された。また、類似部品と被類似部品が共に 0 個となったヘッダファイルは 16 件であった。部品間の類似度が 1.0 となったソフトウェア部品があったが、同じクラスが二重に宣言されていることがわかった。

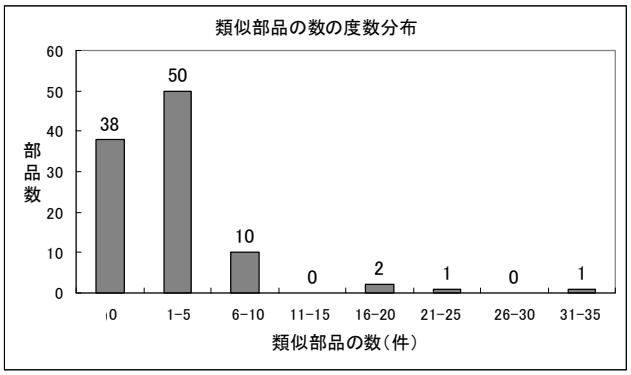

図 2　類似部品数の度数分布

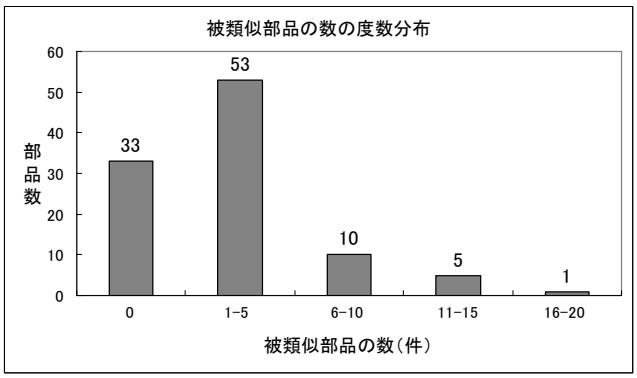

図 3　被類似部品の度数分布

4.3.2 別ライブラリを利用した類似部品の抽出

結果を表 10 に示す。表 1 では、対応する 2 つの部品が類似関係にある場合に“○”を記載している。部品 P と類似関係にある部品とは、部品 P の類似部品と被類似部品を合わせたものである。

4.3.3 熟練技術者により選出された類似部品との比較

ここでは、評価ライブラリで開発経験のある技術者が選出した類似関係にある部品と Collate++が抽出した類似関係にある部品を比較した結果を示す。表 8 に熟練技術者それぞれが判定した類似関係にある部品の組合せ総数と、2 名の技術者が共に類似関係にあると判定した部品の組合せ総数を示す。表 9 に熟練技術者 2 名が共に選出した類似関係にある部品の組合せの総数と、そのうち、Collate++が類似関係にある判定した部品の組合せの総数を示す。

表 8　類似関係にある部品に判定された部品の件数

技術者 A による判定	206 組
技術者 B による判定	237 組
技術者 A と技術者 B の両方に判定された部品	83 組

表 9　人手による類似判定との対比

技術者 A と B の両方により判定された類似関係にある部品の総数	83 組
そのうち Collate++が検出できた類似関係にある部品の総数	46 組

4.4 考察

4.4.1 抽出された類似部品の数

まず、類似部品数の多い部品の特徴を考察する。類似部品数が 15 以上となったソフトウェア部品 4 件には、別のヘッダファイルのクラスを継承するクラスが宣言されており、そのメンバ関数は仮想関数のオーバーライドが多く、独自の関数が少なかった。さらに、独自の関数には、Add や Copy など単純で短い名前の関数が多かった。このように、独自の機能を持たない部品に類似部品が多くなったが、これらの類似部品をユーザに提示しても有益な情報は得られないと考えられる。これを防ぐためには、ソースコード中でよく用いられ、特定の機能を表現していない Add や Copy などの単語を除外して類似度を求める方法が考えられる。

次に、被類似部品の数が多い部品について考察する。被類似部品数が上位 5 位の部品のうち 4 件の部品は、ファイル上部で宣言されたクラスを継承するクラスが同じファイルに複数宣言されていた。例えば、最も被類似部品の多い部品では、ファイル上部に GUI 部品用の抽象クラスが宣言され、以降でボタンやフィールドなど、機能の異なるクラスが 13 個宣言されている。この場合、類似部品として選出されても、どの機能について類似性が見られたかをユーザが判断しにくいため、あらかじめ機能がより明確な単位をソフトウェア部品とするのがよい。また、残り 1 件の部品には、ビュー機能の基底クラスが宣言されていたが、この部品が多くの部品の類似部品として検出されることは提案手法の狙いと合致している。

さらに、類似部品と被類似部品が共に 0 個となった 16 個のソフトウェア部品については、csv ファイルの入出力機能やログ機能など、他のソフトウェア部品とは独立した機能をもつものが多く、類似部品がないことは妥当といえる。ただし、マウスドラッグ操作を実現するクラスからなるソフトウェア部品も、他の部品も同様の機能をもつにもかかわらず類似部品が検出されなかった。これは、他の部品がドラッグ機能を持っていても他の機能を果たす API が多いため、類似部品と判定されなかったと考えられる。このことは、ソフトウェア部品の単位と、各種しきい値の与え方についてさらなる検討が必要なことを示している。

4.4.2 別ライブラリを利用した類似部品の抽出

表 10 より、評価ライブラリのトグルボタン A と別ライブラリのラジオボタン、評価ライブラリのラジオボックス A と別ライブラリのトグルボックスが類似関係にあると判定された。これらの部品は、いずれもボタンを押すことにより、部品内の状態遷移が行なわれる点で類似しているといえるため、判定結果は妥当と見なせる。

その一方、評価ライブラリのボタン部品と別ライブラリのボタン機能をもつ部品、評価ライブラリの文字列部品と別ライブラリの文字列表示機能のある部品など、明らかに類似性の見られる部品同士を類似関係にあると判定できない場合も見られた。

評価ライブラリと別ライブラリで比較した部品は、それぞれ継承関係における関連は小さいため、部品中で宣言されるクラスの継承関係だけで、これらの部品の類似性を判定することは難しい。そのため、トグルボタンやラジオボックスなどの部品を類似関係にあると検出できたことは、提案手法により継承関係や部品名だけでは抽出できない類似部品を抽出できることを示唆する。ただし、明らかに類似関係の見られる部品同士を類似部品と判定できない場合も多く見られたため、類似度の算出方法に調整が必要なことがわかる。

表 10　別ライブラリの類似部品

		評価ライブラリの部品									
		ボタン部品	フィールド部品	アイコン	ラベル部品	リスト部品	オプションメニュー	セパレータ	文字列部品	ラジオボックス A	トグルボタン A
別ライブラリの部品	数値フィールド	○	○		○				○	○	○
	数値ボタン										○
	時間表示ボタン	○									
	帳票用ボタン										
	テキストボックス								○		○
	ラジオボックス B									○	
	ラジオボタン										○
	トグルボックス									○	
	トグルボタン B										○

4.4.3 熟練技術者により選出された類似部品との比較

Collate++で抽出できた類似関係にある部品は、2 名の技術者が共に選出した部品の 55%であった。選出した技術者より、“同時に使われることの多いクラスも類似と判定した”、“クラスの継承関係のある部品を選択した”とのコメントがあったが、Collate++では、これらの関係性をもつ部品間の類似度が必ずしも高くなるように算出されない。技術者がもつ類似性の感覚と合わせるには、クラスの依存関係や継承関係を表現するメトリクスの併用を検討することが求められる。

ただし、双方の技術者が共に類似していると判定した部品は、それぞれの技術者が選出した類似部品の件数の半数以下であり、類似性の判定基準は技術者間でも差異が大きいことがわかる。これは、技術者がどのような観点で類似性を判定するかについて、調査が必要なことを示している。

5. まとめ

本稿では、ソフトウェア部品を再利用する場面において、所望の機能を表すキーワードで部品を検索する際にキーワードの表記揺れや不足による悪影響を軽減する方法を検討した。具体的には、キーワードによる検索に加え、キーワードが対応付けられたソフトウェア部品と機能が類似する部品も候補として提示する方法を検討した。

提案手法では、ソフトウェア部品が提供する公開 API が部品の機能を表していると仮定し、API 中の型と識別子の類似度に基づいて、ソフトウェア部品間の類似度を算出する。

また、自社製 C++クラスライブラリを用いて、部品間の類似度の算出し、その結果を考察した。その結果、クラスの継承関係を超えて同じ機能をもつ GUI 部品が選出されるなど、提案手法によって類似部品を検出できることが示唆された。一方、ライブラリを熟知した技術者がもつ類似性の感覚とのずれもみられたため、他のメトリクスとの併用などを検討していく必要があることがわかった。また、ソフトウェア部品として扱う成果物の単位や、類似部品の判定に用いるしきい値を適切に定めるためには、さらなる評価と検討が必要であることがわかった。

今後は、オープンソースのクラスライブラリなど、他のライブラリを用いてさらに類似度の算出方法の評価を行なっていき、類似度の算出方法の改善を図っていくことを予定している。

参考文献

1) Meiling, R., et al.: Storing and Retrieving Software Components: A Component Description Manager, Proc. Australian Computer Science, pp.107-117(2000).
2) 鷲崎弘宜, 深澤良彰:有向置換性類似度に基づくコンポーネント検索方式の実現と評価, 情報処理学会論文誌, Vol.43, No.6, pp.1638-1652 (2002).
3) 島田隆次, 市井誠, 早瀬康裕, 松下誠, 井上克郎:開発中のソースコードに基づくソフトウェア部品の自動推薦システム A-SCORE, 情報処理, Vol.50, No.12, Page.3095-3107 (2009).
4) 横森励士, 梅森文彰, 西秀雄, 山本哲男, 松下誠, 楠本真二, 井上克郎: Java ソフトウェア部品検索システム SPARS-J, 電子情報通信学会 D-I, Vol.J87-D-I, No.12, pp.1060-1068, (2004).
5) 粂照宣, 鵜飼孝典, 三末和男: プロジェクト資産再利用インタフェース, 情報処理学会研究報告, GN, 2003(106), pp.139-144, (2003).
6) 山本哲男, 松下誠, 神谷年洋, 井上克郎: ソフトウェアシステムの類似度とその計測ツール SMMT, 電子情報通信学会論文誌 D-I, vol. J85-D-I, No.6, pp.503-511, (2002).
7) Michail, A., et al.: Assesing Software Libraries by Browsing Similar Classes, Functions and RelationShips, Proc. International Conference Software Engineering, Vol.21, pp.463-472(1999).
8) Google: Google C++ Style Guide, http://google-styleguide.googlecode.com/svn/trunk/cppguide.xml (2012).
9) Microsoft: Coding Techniques, http://msdn.microsoft.com/en-us/library/Aa291593 (2012).