Vol.2019-SE-202 No.14
2019/7/13

API利用パターン情報抽出のためのイベント駆動型アプリケーションにおけるオーバライドされたイベントリスナの解析手法の検討

西山 佳志[1]　西本 匡志[1]　近藤 悠志[1]　川端 英之[1]　弘中 哲夫[1]

概要：スマホアプリの開発においては，予め提供されるフレームワークのクラスを継承した独自定義のサブクラスの中で標準的メソッドをオーバライドして使用することが頻繁に行われる．オーバライドされて使用されるメソッドの組み合わせに関する情報は，API 利用パターン情報の一種であり，コード記述時のプログラマへの推薦やソフトウェアの検証等，アプリケーション開発支援に活用できる可能性がある．同時に生じる事象の関係を分析する手法としてはバスケット分析が知られているが，その単純な適用は，オーバライドされるメソッドの組み合わせの分析には十分ではない．我々は，オープンソースリポジトリマイニングで得られる統計情報に基づいてオーバライドされるメソッド同士の関係の分析におけるユールの連関係数の有効性について調査した．調査結果からは，単純なバスケット分析ではなくユールの連関係数に基づくフィルタリングが有効であることが示唆される．本稿では，調査結果をまとめ，オーバライドされるメソッド同士の関係を踏まえた推薦機能や誤り検出機能の実現に向けて考察する．

キーワード：API，オーバライドメソッド，推薦，ユールの連関係数，Android

A Study of Analysis Method of Overridden Event Listeners in Event-Driven Application for Extracting API Usage Pattern Information

Keiji Nishiyama[1]　Masashi Nishimoto[1]　Yushi Kondoh[1]　Hideyuki Kawabata[1]
Tetsuo Hironaka[1]

Abstract: In smartphone application development, some standard methods are frequently overridden and used in subclasses of custom-defined classes that inherit from framework classes provided in the environment. The information on the combination of methods used by being overridden is a kind of API usage pattern information, and it can be used for application development support such as recommendation to programmers and verification of software. Although affinity analysis is known as a method to analyze the relationship of events that occur simultaneously, its simple application is not sufficient for analysis of combinations of methods to be overridden. We investigated the effectiveness of Yule's coefficient of association in the analysis of relationships between methods that are overridden based on statistical information obtained in open source repository mining. From the survey results, it is suggested that filtering based on Yule's coefficient is effective rather than simple affinity analysis. In this paper, we summarize the survey results and consider for realization of the recommendation and the error detection functionalities based on the relationship between overridden methods.

Keywords: API, Overridden methods, Recommendation, Yule's coefficient of association, Android

1. はじめに

イベント駆動型アプリケーションの典型ともいえるスマホアプリは，ライフサイクルに従ったイベント処理のほか，多種多様なデバイスからの入力に応答するためのイベント処理の記述の組み合わせから成る基本構造を持っている．そして，スマホアプリを構成する各種機能は，個々のイベント処理の中に組み込む形で実装される．従って，スマホアプリの開発においては，予め提供されるフレームワークのクラスを継承した独自定義のサブクラスの中で標準的メソッドをオーバライドして使用することが頻繁に行われる．例えば，Android フレームワークにおけるクラス Activity はそのような利用のされ方をするクラスの一つである．

フレームワークで提供される各種クラスに含まれるメソッドには，一つのアプリケーションプログラムの中で同時にオーバライドされる頻度が高いものがある．例えば，Android のクラス Activity においては，各種デバイスの処理開始と処理終了を制御するメソッドとして onResume および onPause の組み合わせの使用頻度が極めて高い（実際，この組み合わせでのアプリケーション記述が推奨されている）．オーバライドされて使用されるメソッドの組み合わせに関する情報は，API 利用パターン情報の一種であり，コード記述時のプログラマへの推薦やソフトウェアの検証等，アプリケーション開発支援に活用できる可能性がある．しかしながら我々の知る範囲ではこれらに関する研究は見られない．

同時に生じる事象の関係を分析する手法としては，バスケット分析が広く知られている．分析結果の応用は，消費者に対する商品の推薦機能などの形でなされている．我々は，バスケット分析の手法を，オーバライドされて使用されるメソッドそれぞれの組み合わせの分析に適用した．そこで得られた知見は，単純なバスケット分析手法はオーバライドされるメソッドの組み合わせの分析には十分ではないということである．すなわち，メソッドの推薦に当たっては個々の組み合わせの出現頻度だけでなく機能的な関係を踏まえる必要があり，個々の商品の関連を深く追求する必要のない商品推薦の文脈を単純に当てはめることができないということである．しかしながら，オーバライドされるメソッド同士の機能的な関係の分析は容易ではない．

我々は，Android アプリケーション開発支援を対象とした推薦機能や誤り検出機能の実現に役立つ情報を得ることを目的とし，オープンソースリポジトリマイニングで得られる統計情報に基づいてオーバライドされるメソッド同士の関係の分析におけるユールの連関係数の有効性について調査した．調査結果からは，単純なバスケット分析ではなくユールの連関係数に基づくフィルタリングが有効であることが示唆される．本稿では，調査結果をまとめ，オーバライドされるメソッド同士の関係を踏まえた推薦機能や誤り検出機能の実現に向けての考察を行う．

以下，2 章では，二項関係の相関を抽出する 2 つの手法について説明する．3 章では，同時にオーバライドされる頻度が高いメソッドの二項関係を定義する．4 章では，大量の Android アプリケーションから抽出して分析した二項関係のうち，いくつかの特徴的な例を提示する．5 章では，4 章で得られた二項関係について，オーバライドメソッドの推薦に対する有用性を考察する．6 章では，関連研究を述べ，7 章でまとめる．

[1] 広島市立大学
Hiroshima City University

2. 準備：オーバライドメソッド間の二項関係の分析手法

我々はオーバライドメソッド間の相関関係を抽出する ために，バスケット分析とユールの連関係数の 2 つの分析手法を用いる．本節ではこれらの 2 つの分析手法について説明する．

2.1 バスケット分析

バスケット分析は，大量のアイテム集合からアイテム間の相関ルール (association rules) を抽出する手法である．例えば，「アイテム A が出現したとき，アイテム B も出現する」という関係を，相関ルール A→B で表す．このとき，矢印の左側を条件部，右側を結論部と呼ぶ．バスケット分析では，相関ルールの重要度を示す評価指標として，支持度 (Support)，確信度 (Confidence)，リフト値 (Lift) が用いられる．なお，以下では，アイテム A および B が同時に出現する集合の集合を $A \wedge B$，アイテム A が出現しかつアイテム B が出現しない集合の集合を $A \wedge \overline{B}$ などと表記する．

- **支持度**
  アイテム A の支持度 supp (A) は，全アイテム集合のうち，アイテム A を含むアイテム集合の占める割合である．したがって，支持度 supp(A→B) は，全アイテム集合のうち，アイテム A と B の組み合わせを含むアイテム集合の占める割合である．
  $$supp(A \to B) = \frac{N(A \wedge B)}{N(U)} = supp(B \to A) \quad (1)$$
- **確信度**
  確信度 conf(A→B) は，アイテム A を含むアイテム集合のうち，アイテム B も同時に含むアイテム集合の占める割合である．すなわち，「アイテム A が出現したとき」という条件下でのアイテム B の支持度である．
  $$conf(A \to B) = \frac{N(A \wedge B)}{N(A)} \quad (2)$$
- **リフト値**
  リフト値は，確信度 conf(A→B) を結論部の支持度で

割った値である．リフト値は，アイテム B 単独での出現確率に対して，「アイテム A と B が同時に出現したとき」という条件を加えた場合の比率を表している．リフト値が 1 より大きいとき，アイテム A を含むアイテム集合中のアイテム B の出現確率は，アイテム B 単独での出現確率より大きいことを表す．

$$lift(A \to B) = \frac{conf(A \to B)}{supp(B)} \tag{3}$$

2.2 ユールの連関係数

ユールの連関係数は，大量のアイテム集合からアイテム間の相関の強さを抽出する手法である．バスケット分析との違いは，双方向の関係を踏まえた指標であるという点である．ユールの連関係数は，1 に近づくに連れ X↔Y が強い相関になることを表している．

$$Q = \frac{ad - bc}{ad + bc} \tag{4}$$

なお，ここで，$a = N(A \wedge B)$，$b = N(A \wedge \overline{B})$，$c = N(\overline{A} \wedge B)$，$d = N(\overline{A} \wedge \overline{B})$ である．

3. オーバライドメソッドの推薦に向けた二項関係の分析手法

3.1 概要

本稿の目的は，メソッド A をオーバライドしている人に，メソッド B のオーバライドも推薦することである．例えば，図 1 のソースコードにおいて，Activity クラスの onCreateOptionMenu をオーバライドしている人に，Activity クラスの onOptionsItemSelected のオーバライドを推薦することである．onCreateOptionMenu は Android の端末画面にメニューが表示される際に呼び出されるイベントリスナであり，独自のオプションメニューに応じた処理を実装できる．一方，onOptionsItemSelected はユーザによってメニューから項目が選択された際に呼び出されるイベントリスナであり，選択された項目に応じて処理を実装できる．これらの 2 つのイベントリスナはメニューの表示や選択に深く関わるものであり，頻繁かつ同時にオーバライドされるべきものである．

メソッドのオーバライドに関わる使用パターンに関する情報は，アプリケーションの開発者にとって有用なものであると考えられるが，オーバライドメソッドに関する推薦に関する手法は推薦されていない．

オーバライドメソッドに関する推薦を実現するために，オーバライドメソッド A が記述されているときに，オーバライドメソッド B が記述されているといった相関関係 (A→B) を把握する必要がある．そこで，我々はオープンソースリポジトリに含まれるソースファイルからオーバライドメソッドの相関関係を集めて，推薦に利用することを考えている．相関関係の分析手法として，よく知られているバスケット分析の指標のみでは，オーバライドメソッドに関する推薦に適した関係を把握するのには不十分であった．我々は，バスケット分析の指標に加えてユールの連関係数を使用すれば，オーバライドメソッドに関する推薦に有効な相関関係が把握できると期待している．

```
public class MainActivity extends Activity
                implements View.OnClickListener {
   ...
   @Override
   protected void onCreate(Bundle savedInstanceState) {
       super.onCreate(savedInstanceState);
       setContentView(R.layout.sample_main);
   }
   @Override
   public boolean onCreateOptionsMenu(Menu menu) {
       getMenuInflater().inflate(R.menu.main, menu);
       return true;
   }
   ...
}
```

図 1 Android のオプションメニューに関するプログラム記述例

3.2 オーバライドメソッドの相関関係に関する定義

本稿で抽出したいオーバライドメソッドの相関関係は，次の通りである．

- Android クラスのサブクラスにおける，全てのオーバライドメソッド間の二項関係が対象である．
- オーバーロードは個別に集計する．
- インナークラスは別個集計する．
- 片方向の共起関係である．つまり，オーバライドメソッド A がオーバライドメソッド B に連動して実装される可能性が高いことを表す（この場合，B → A に連動して実装される可能性は考慮しない）．
- 複数階層に渡った継承の場合は，最下層のクラス（Android クラスのサブクラスの 1 階層上）として集計をする．

4. 評価：データ分析

我々は，メソッド A をオーバライドしている人にメソッド B のオーバライドを推薦するために，大量の Android プロジェクトに対して推薦に利用可能な相関関係があるかどうかを明らかにする目的でデータ分析を行った．

4.1 分析に用いた Android アプリケーション

Github に登録されている Android アプリケーションを対象として，オーバライドメソッド間の二項関係に関する分析を行った．本分析に用いたプロジェクトは，スター数が 100 以上の Android アプリケーションのプロジェクト 2,279 個のうち，各プロジェクトに含まれる Java ファイル数が 100 以下の 1,810 個である．総 Java ファイル数は

48,401 個である．

4.2 オーバライドメソッドの相関関係に関する抽出手順

Android アプリケーションの Java ファイルからオーバライドメソッドの相関関係を抽出する手順について述べる．まず，Eclipse JDT[*1]の構文解析器を用いて，Java ファイルに含まれる Android クラスのサブクラスとその中で定義されているメソッドに関する情報を抽出する．次に，各サブクラスに定義されているオーバライドメソッド N の中から 2 つのメソッドを選ぶ組み合わせ $({}_nC_2)$ を作成する．例えば，10 個のオーバライドメソッドが定義されたサブクラスの場合，オーバライドメソッドの二項関係が ${}_{10}C_2$ で 45 個得られる．最後に，得られた全てのオーバライドメソッドの二項関係に対して，バスケット分析，および，ユールの連関係数を求める．

4.3 分析結果

2,279 個の Android プロジェクトには，Android クラスのサブクラスが計 34,100 個，オーバライドメソッドが計 66,311 個，オーバライドメソッド間の二項関係が計 95,495 個存在した．

いくつかのオーバライドメソッドを条件部とした，それぞれの二項関係を表 1〜3 に示す．なお，確信度 0.005 以下の二項関係については省略している．各表には，Android のドキュメントに掲載されている情報から読み取った，推薦することが有用だと考えられる二項関係がそれぞれ一組ずつ含まれている．これらの二項関係を以後，推薦すべき二項関係と呼ぶ．

表 1〜3 の全ての推薦すべき二項関係について，ユールの連関係数は全て 0.9 以上の高い指標を示している．一方で確信度は 0.381〜1.000 と幅広い．このことから，有用な二項関係であるか否かはユールの連結係数と強い相関がある一方で，確信度との相関は弱いといえる．よって，例えば (i) ユールの連関係数 ≥ 0.9 の二項関係が推薦に有用な可能性がある．

また推薦すべき二項関係以外について，ユールの連関係数が 0.9 以上の二項関係あるが，それらの確信度は全て 0.1 以下の低い指標を示しており，推薦には十分ではない可能性がある．よって，(ii) 確信度は推薦の強さの指標として有用な可能性がある．

5. 考察：推薦ツールへの組み込み時の効果について

推薦ツールへの組み込みにあたって，以下のルールにより，推薦すべき二項関係を抽出可能であると考えられる．

- (i) 全ての二項関係について，ユールの連関係数 ≥ 0.9 の二項関係を選択する．
- (ii) (i) で選択した二項関係について，ユールの連関係数と確信度の指標を用いて推薦の強さを導く．

分析によって得られた二項関係を利用した推薦ツールを想定して，有用な推薦が行えるかどうかを評価する．

[*1] https://www.eclipse.org/jdt/

```
public static class SampleFragment extends Fragment {
    OnArticleSelectedListener mListener;
    ...
    @Override
    public void onAttach(Activity activity) {
        super.onAttach(activity);
        try {
            mListener =
             (OnArticleSelectedListener)activity;
        } catch (ClassCastException e) {
            throw new ClassCastException(
              activity.toString() + " must ...");
        }
    }
    ...
}
```

図 2 Fragment においてイベントを Activity と共有するソースコード例

5.1 Fragment: onAttach→onDetach

フラグメントは，アクティビティと関連付けられると onAttach，解除されると onDetach が呼ばれる．イベントをアクティビティと共有したい場合は，図 2 のようにフラグメント内でコールバックインターフェースを定義し，onAttach 内でアクティビティと関連付けることがドキュメントで推奨されている．一方で，解除方法については言及がなく，多くのユーザは onDetach 内で解除の処理を実装している．

ドキュメントに従って図 2 のソースコードを記述した後に, onAttach と同時にオーバライドが推薦されているメソッドを調べるとする．表 1 において，onAttach→onDetach のユールの連関係数が 0.902 で突出した高い指標を示しており，確信度は 0.381 と低いが，onDetach が推薦されることを知る．よって，イベントの関連付けを解除する必要があることに気が付くことができる可能性がある．

5.2 ViewPager: onTouchEvent→onInterceptTouchEvent

ViewPager は，ページを横スライドで切り替えられるビューである．指でページを横スライドできる機能が標準で備わっているが，しばしばそれが不要な場合がある．例えば，ボタンの押下でのみスライドを許可したい場合などである．それには，図 3 のソースコード例のようにサブクラスでメソッド onTouchEvent とメソッド onInterceptTouchEvent をオーバライドし，共にリターン文で false を返すことで，スライドを許さないように変更する実装パ

表 1 オーバライドメソッド（A: onAttach）の二項関係に関する分析結果

B（メソッド名）	支持度 (A∧B)	確信度 (A→B)	確信度 (B→A)	リフト値 (A→B)	リフト値 (B→A)	ユールの連関係数	推薦すべき二項関係
onDetach	0.030	0.381	0.511	6.590	4.916	0.902	○
setUserVisibleHint	0.006	0.071	0.243	3.139	0.922	0.604	×
onStart	0.009	0.111	0.226	2.914	1.434	0.582	×
onStop	0.009	0.111	0.206	2.657	1.434	0.540	×
onActivityResult	0.007	0.095	0.182	2.346	1.229	0.476	×
onResume	0.020	0.254	0.159	2.055	3.277	0.457	×
onPause	0.011	0.143	0.157	2.020	1.844	0.414	×
onCreate	0.034	0.437	0.131	1.686	5.633	0.412	×
onCreateOptionsMenu	0.010	0.127	0.157	2.024	1.639	0.410	×
onDestroy	0.010	0.127	0.113	1.454	1.639	0.227	×
onOptionsItemSelected	0.006	0.079	0.106	1.373	1.024	0.185	×
onDestroyView	0.010	0.127	0.099	1.275	1.639	0.149	×
onActivityCreated	0.014	0.175	0.084	1.084	2.253	0.052	×
onViewCreated	0.018	0.230	0.076	0.982	2.970	-0.013	×
onCreateView	0.068	0.873	0.077	0.990	11.266	-0.045	×

表中のメソッドは `android.support.v4.app.Fragment` クラスに属している．

表 2 オーバライドメソッド（A: onSaveInstanceState）の二項関係に関する分析結果

B（メソッド名）	支持度 (A∧B)	確信度 (A→B)	確信度 (B→A)	リフト値 (A→B)	リフト値 (B→A)	ユールの連関係数	推薦すべき二項関係
getSuggestedMinimumHeight	0.006	0.086	1.000	13.672	1.179	1.000	×
getSuggestedMinimumWidth	0.006	0.086	1.000	13.672	1.179	1.000	×
onRestoreInstanceState	0.073	1.000	0.983	13.441	13.672	1.000	○
setOnClickListener	0.005	0.069	0.444	6.077	0.943	0.831	×
performClick	0.005	0.069	0.400	5.469	0.943	0.800	×
onMeasure	0.062	0.845	0.137	1.871	11.551	0.765	×
onTouchEvent	0.050	0.690	0.188	2.568	9.429	0.757	×
onVisibilityChanged	0.006	0.086	0.238	3.255	1.179	0.618	×
setBackgroundColor	0.005	0.069	0.211	2.878	0.943	0.561	×
draw	0.005	0.069	0.200	2.735	0.943	0.538	×
onAttachedToWindow	0.011	0.155	0.170	2.322	2.122	0.485	×
onSizeChanged	0.035	0.483	0.123	1.679	6.601	0.428	×
onDetachedFromWindow	0.016	0.224	0.144	1.975	3.065	0.423	×
onDraw	0.068	0.931	0.076	1.038	12.730	0.232	×
onLayout	0.008	0.103	0.074	1.013	1.414	0.008	×

表中のメソッドは `android.view.View` クラスに属している．

ターンがよくみられる．しかし，View クラスから頻繁にオーバライドされる onTouchEvent に比べ，あまり認知度が高いとはいえない onInterceptTouchEvent のオーバライドを忘れてしまう可能性がある．

onTouchEvent をオーバライドして図 3 と同様の記述を行った後に，オーバライドが推薦されるメソッドを調べるとする．表 3 において，ユールの連関係数が高い指標 (例えば 0.9 以上) を示す二項関係は複数見られるが，onTouchEvent→onInterceptTouchEvent の確信度が数が 0.971 で突出して高い指標を示しており，onInterceptTouchEvent が推薦されることを知る．それにより，onInterceptTouchEvent のオーバライドの必要性に気が付くことができる可能性がある．

6. 関連研究

オープンソースリポジトリマイニング等によって再利用し易い情報を抽出し，ユーザへの推薦やプログラムの違反検出等に活用する研究がなされている．早瀬ら [1] は，特定のユーザ定義メソッド内で使用される可能性が高い API メソッドの集合を抽出し，開発者に推薦するツールを提案している．あるユーザ定義メソッドを構成する単語とそのメソッド内で使用される可能性がある API メソッド集合を対応づけたデータベースを生成する際には，我々と同様，バスケット分析手法の 1 つである，相関ルールマイニングを使用している．Android アプリケーションの開発を支援する DroidAssist[2] は，隠れマルコフモデルに基づいた API

表 3 オーバライドメソッド（A: onTouchEvent）の二項関係に関する分析結果

B（メソッド名）	支持度 (A∧B)	確信度 (A→B)	確信度 (B→A)	リフト値 (A→B)	リフト値 (B→A)	ユールの連関係数	推薦すべき二項関係
requestDisallowInterceptTouchEvent	0.009	0.015	1.000	1.721	0.025	1.000	×
onSaveInstanceState	0.009	0.015	1.000	1.721	0.025	1.000	×
fakeDragBy	0.009	0.015	1.000	1.721	0.025	1.000	×
canScrollVertically	0.009	0.015	1.000	1.721	0.025	1.000	×
canScrollHorizontally	0.009	0.015	1.000	1.721	0.025	1.000	×
performClick	0.009	0.015	1.000	1.721	0.025	1.000	×
getChildDrawingOrder	0.017	0.029	1.000	1.721	0.051	1.000	×
setOnFocusChangeListener	0.026	0.044	1.000	1.721	0.076	1.000	×
setOnTouchListener	0.026	0.044	1.000	1.721	0.076	1.000	×
onInterceptTouchEvent	0.556	0.956	0.844	1.452	1.645	0.971	○
scrollTo	0.026	0.044	0.750	1.290	0.076	0.378	×
onRestoreInstanceState	0.017	0.029	0.667	1.147	0.051	0.185	×
setCurrentItem	0.077	0.132	0.529	0.911	0.228	-0.123	×
onSizeChanged	0.009	0.015	0.500	0.860	0.025	-0.165	×
setCurrentItem	0.060	0.103	0.467	0.803	0.177	-0.259	×
onMeasure	0.051	0.088	0.400	0.688	0.152	-0.399	×
setAdapter	0.051	0.088	0.375	0.645	0.152	-0.452	×
onAttachedToWindow	0.009	0.015	0.333	0.574	0.025	-0.481	×
getCurrentItem	0.026	0.044	0.333	0.574	0.076	-0.503	×
onPageScrolled	0.009	0.015	0.250	0.430	0.025	-0.628	×
setPageTransformer	0.009	0.015	0.250	0.430	0.025	-0.628	×
dispatchDraw	0.009	0.015	0.250	0.430	0.025	-0.628	×
setOnPageChangeListener	0.017	0.029	0.250	0.430	0.051	-0.643	×
getAdapter	0.009	0.015	0.143	0.246	0.025	-0.807	×
dispatchTouchEvent	0.009	0.015	0.125	0.215	0.025	-0.836	×

表中のメソッドは `android.support.v4.view.ViewPager` クラスに属している．

```
public class SampleViewPager extends ViewPager {
    private boolean isPagingEnabled;
    ...
    @Override
    public boolean onTouchEvent(MotionEvent event) {
        if (this.isPagingEnabled) {
            return super.onTouchEvent(event);
        }
        return false;
    }
    @Override
    public boolean onInterceptTouchEvent(MotionEvent event) {
        if (this.isPagingEnabled) {
            return super.onInterceptTouchEvent(event);
        }
        return false;
    }
    ...
}
```

図 3 ViewPager において指の横スライドの許可を制御するソースコード例

メソッドの推薦システムである．PR-Miner[3] は，オープンソースから抽出された API メソッドの利用パターンに対し頻出アイテムセットマイニングを適用することで，文書化されていない暗黙のプログラミングルールを生成し，プログラムの違反検出に活用する．FrUiT[4] は，association rule mining を用いて，オーバライドメソッドに関する情報を再利用可能なパターンとして抽出する．LibraryGuru[5] は，Android アプリケーションにおける特定の機能を実装するために使用すべき API メソッドのまとまりと同時に，その機能を実装するために必要となるオーバライドメソッドの推薦も行う．これらの研究からも分かるように，オーバライドメソッドを含めた API 利用パターン情報を API メソッドの推薦やプログラムの違反検出に活用することは有意義であると考えられる．

7. おわりに

本稿では，オープンソースリポジトリマイニングで得られる統計情報に基づいてオーバライドされるメソッド同士の関係の分析におけるユールの連関係数の有効性についての調査結果について述べた．単純なバスケット分析ではなくユールの連関係数に基づくフィルタリングが有効であると考えられる．また本稿では，オーバライドされるメソッド同士の関係を踏まえた推薦機能について，いくつかのシナリオを想定して考察した．今後の課題として，より多くの Android アプリケーションに関する調査のほか，オーバ

ライドメソッド内に記述された API メソッドを考慮した推薦機能の実現などが挙げられる．

参考文献

[1] 早瀬康裕，鬼塚勇弥，山本哲男，　石尾隆，井上克郎：API 呼び出しとメソッド周辺の識別子の実績に基づいた API 集合推薦手法，情報処理学会論文誌，Vol. 56, No. 2, pp. 692–700 (2015).

[2] Nguyen, T. T., Pham, H. V., Vu, P. M. and Nguyen, T. T.: Recommending API Usages for Mobile Apps with Hidden Markov Model, *Proceedings of 30th IEEE/ACM International Conference on Automated Software Engineering (ASE 2015)*, pp. 795–800 (2015).

[3] Li, Z. and Zhou, Y.: PR-Miner: Automatically Extracting Implicit Programming Rules and Detecting Violations in Large Software Code, *Proceedings of the 10th European software engineering conference held jointly with 13th ACM SIGSOFT international symposium on Foundations of software engineering (ESEC/FSE-13)*, pp. 306–315 (2005).

[4] Bruch, M., Schäfer, T. and Mezini, M.: FrUiT: IDE Support for Framework Understanding, *Proceedings of the 2006 OOPSLA Workshop on Eclipse Technology eXchange*, eclipse '06, New York, NY, USA, ACM, pp. 55–59 (online), DOI: 10.1145/1188835.1188847 (2006).

[5] Yuan, W., Nguyen, H. H., Jiang, L. and Chen, Y.: Poster: LibraryGuru: API Recommendation for Android Developers, *2018 IEEE/ACM 40th International Conference on Software Engineering: Companion (ICSE-Companion)*, pp. 364–365 (2018).