

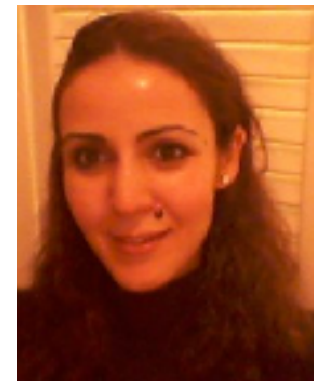
Generating API Call Rules from Version History and StackOverflow Posts



SHAMS AZAD



PETER C. RIGBY



LATIFA GUERROUJ

ACM Transactions on Software Engineering and
Methodology (TOSEM), Volume 25, Issue 4, May 2017 ,
Article No. 29, pp. 1-22

紹介者：宮崎

背景

- API学習はコストがかかる
 - APIの構造を理解しなければならない
 - API進化すれば新しいAPIを覚えなければならない

MediaPlayer.attachAuxEffect
MediaPlayer.getAudioSessionId
⋮
MediaPlayer.getCurrentPosition

Package1つでも
多くのメソッドがあり
覚えるのは大変

- API使用パターンの抽出に関する研究が盛ん
- 抽出したAPI使用パターンを評価するためのベンチマークがない
 - 今まで評価は手作業で行っていた

目的とアプローチ

- 目的
 - API使用パターン抽出手法を評価するためのベンチマークを作成すること
- アプローチ
 - 変更履歴, **StackOverflow**のデータに相関ルールマイニングを適用して, API使用パターン抽出ルールを作成
 - » コミットに含まれるAPIからルールを作成
 - » StackOverflowの回答に含まれるAPIからルールを作成 (ドキュメントからルール作ったのは初!)

なぜ変更履歴とStackOverflow？

- 変更履歴手法
 - 関連する機能を変更履歴から抽出可能[1]
 - 著者らは変更に含まれるAPIからAPI使用パターン抽出ルールを作れると考えた
- StackOverflow手法
 - 著者らはAPIについて議論している投稿を発見
 - 投稿のテキストから似ている投稿を選出可能[2]
 - APIに関する議論からルールを作れると考えた

▲ You can use `MediaPlayer.OnCompletionListener` to listen to the event when a track ends, so that you can play the next one

3 ▼

```
player.setDataSource(path);
player.prepare();
player.setOnCompletionListener(new OnCompletionListener() {

    @Override
    public void onCompletion(MediaPlayer player) {
        player.stop();

        // play next audio file
    }

});
player.start();
```

[1]Thomas Zimmermann, Peter Weisgerber, Stephan Diehl, and Andreas Zeller. Mining version histories to guide software changes. IEEE Transactions on Software Engineering Vol. 31, Issue 6, June 2005, pp. 429-445.

[2]Luca Ponzanelli, Gabriele Bavota, Massimiliano Di Penta, Rocco Oliveto, and Michele Lanza. 2014. Mining Stack Overflow to turn the IDE into a self-confident programming prompter. In Proceedings of the 11th Working Conference on Mining Software Repositories (MSR'14), pp. 102-111

Research Approach

- RA1：変更履歴手法
 - RA1.1：アプリ毎にパターン抽出 ([1]を応用, ベースライン)
 - RA1.2：複数のアプリからパターン抽出
 - RA1.3：アプリのカテゴリ (Ex. Business, Finance) 毎にパターン抽出
- RA2：StackOverflow手法
 - RA2.1：プラス投票された回答に含まれるAPI群からパターン抽出
 - RA2.2：プラス投票された回答のコードスニペットのAPI群からパターン抽出
 - RA2.3：プラス投票された回答の自由記述文に含まれるAPI群からパターン抽出
- RA3：RA1.2とRA2.1の組み合わせ手法

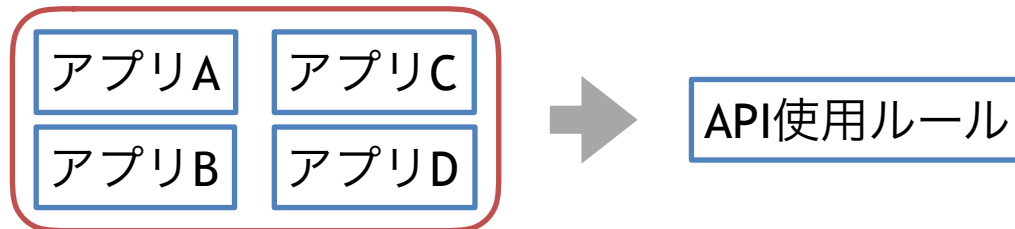
[1]Thomas Zimmermann, Peter Weisgerber, Stephan Diehl, and Andreas Zeller. 2005. Mining version histories to guide software changes. IEEE Transactions on Software Engineering 31, 6, 429-445.

変更履歴手法

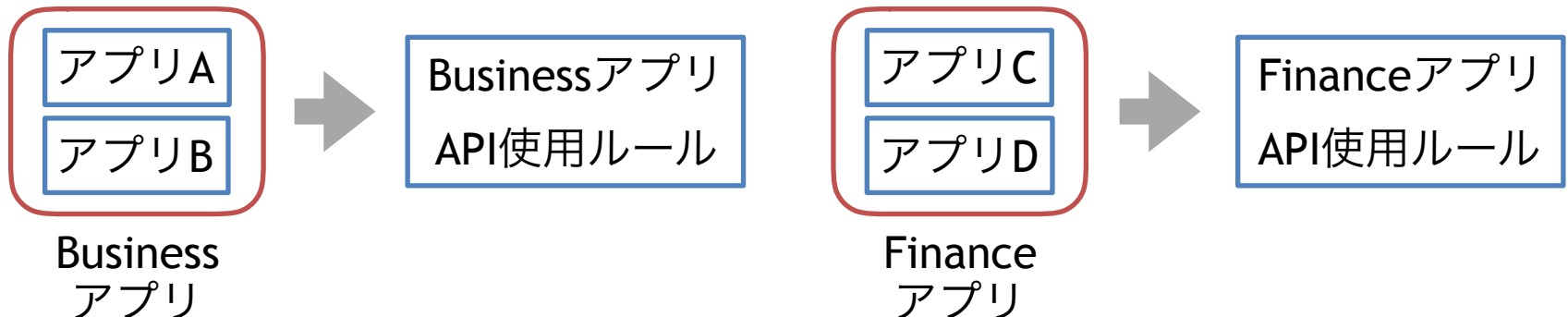
- RA1.1：アプリ毎にルール作成



- RA1.2：複数のアプリの使用パターンからルール作成



- RA1.3：アプリのカテゴリ毎にルール作成



StackOverflow手法

- RA2.1 : プラス投票の回答に含まれるAPI群からパターン抽出
- RA2.2 : プラス投票の回答のコードスニペットのAPI群からパターン抽出
- RA2.3 : プラス投票の回答の自由記述文に含まれるAPI群からパターン抽出

RA2.1

```
3
You can use MediaPlayer.onCompletionListener to listen to the event when a track ends, so that you can play the next one

player.setDataSource(path);
player.prepare();
player.setOnCompletionListener(new OnCompletionListener() {

    @Override
    public void onCompletion(MediaPlayer player) {
        player.stop();

        // play next audio file

    }

});
player.start();
```

RA2.2

RA2.3

share Improve this answer

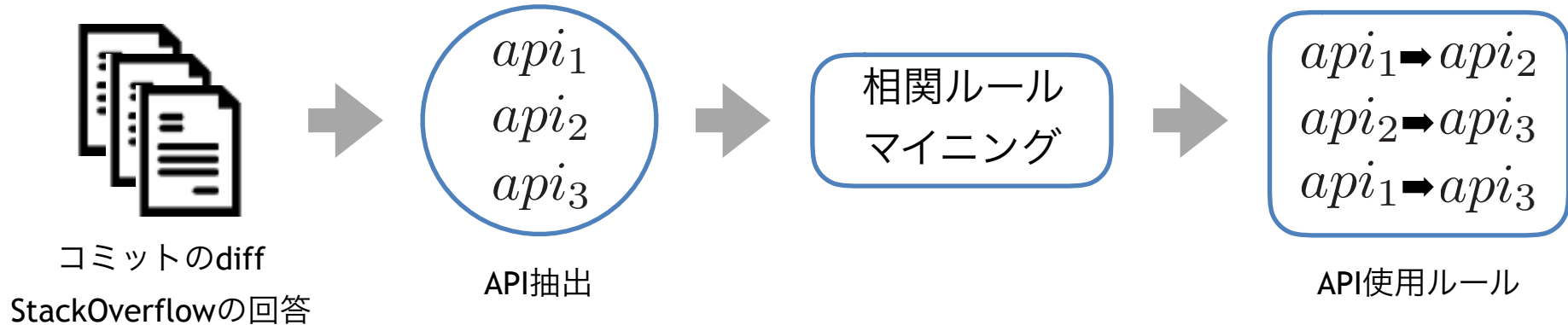
answered Jul 2 '12 at 10:20



prosperK

101k • 23 • 132 • 168

ルール作成の手順



1.APIの抽出

- 変更履歴：1コミットに含まれる追加・削除されたAPI群
- StackOverflow：プラス投票された回答に含まれるAPI群

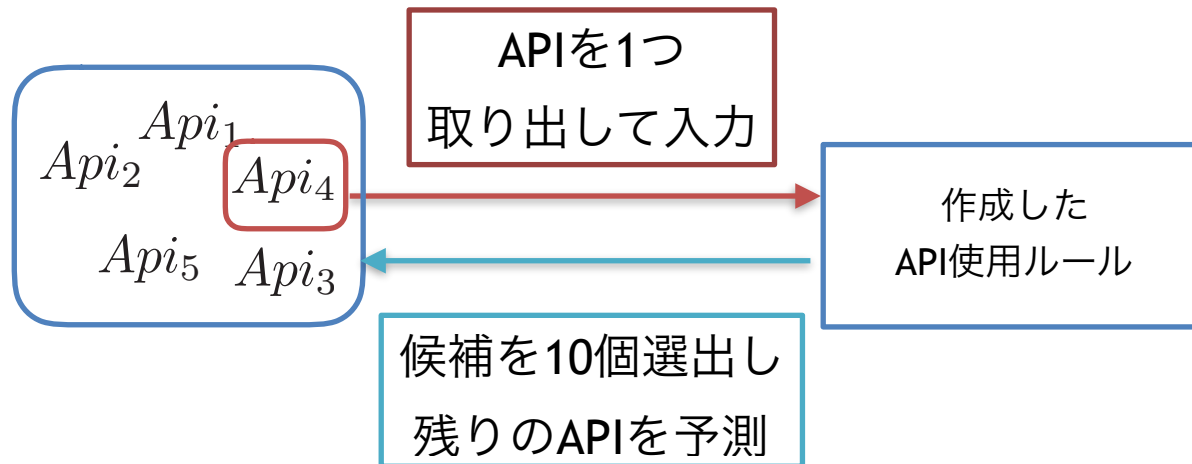
2.相関ルールマイニングを適用し，API使用ルールの作成

- 閾値以上の支持回数と確信度の使用パターンでルールを作成
 - » 支持回数：アプリXとアプリYが一緒に使用されている回数
 - » 確信度：Xが使用されるとYも使用される確率

評価実験

- 変更に使われたAPI使用パターンを予測

- テストデータ・・・直近20%の変更履歴
- 学習データ
 - » 変更履歴モデル・・・残り80%
 - » Stack Overflowモデル・・・回答全て



- 精度をPrecision, Recallで評価
 - マンホイットニーのU検定を行い、各ルールに有意差があるかどうか確認

データセット

- 変更履歴

Table 1. Characteristics of the Studied Apps

App Domains	App Characteristics			
	#Apps	#Versions	#Changed Files	#API Call Changes
Arcade & Card	3	49	37	353
Books & Reference	7	498	496	4,048
Business	2	79	40	694
Communication	9	1,628	1,161	19,060
Education	7	426	203	6,080
Entertainment	6	705	457	6,110
Finance	5	640	658	9,558
Health & Fitness	3	73	35	736
Libraries & Demo	2	22	24	959
Lifestyle	3	61	48	421
Media & Video	6	790	694	7,808
Music & Audio	4	404	295	3,829
News & Magazines	2	166	100	2,188
Personalization	12	776	440	10,619
Photography	2	15	22	188
Productivity	14	419	502	6,024
Puzzles	6	95	84	824
Social	5	426	288	5,180
Tools	120	4,144	3,844	56,930
Transportation	6	109	107	881
Travel & Local	3	79	86	848
Total	230	12,172	10,180	152,624

- StackOverflow

- 対象とした回答

- » 2つ以上APIが含まれている
- » Androidタグが付随
- » プラス投票されている

	RA2.1	RA2.2	RA2.3
対象 期間	2008年8月～2014年10月		
回答数	113,000	84,000	29,000

実験結果：変更履歴ルール

	Precision	Recall
ベースライン (RA1.1)	0.36	0.27
アプリ全部 (RA1.2)	0.75	0.22
カテゴリごと (RA1.3)	0.81	0.28

- RA1.2はベースラインと比べてPrecisionは向上（p値<0.001）したが、Recallは変化なし
- RA1.3はRA1.2と比べてPrecision, Recall共に向上しているが、有意差はなかった
 - Recallのわずかな向上はRA1.2と比べてAPIの組み合わせが少なくなったから
 - 同じカテゴリのアプリは同様のAPI使用ルールを用いている

実験結果：StackOverflowルール

	Precision	Recall
ベースライン	0.36	0.27
回答全て (RA2.1)	0.66	0.13
コードスニペット のみ (RA2.2)	0.63	0.14
自由記述のみ (RA2.3)	0.62	0.17

- ベースラインと比べて、Precisionは向上したがRecallは低下なった（どちらもp値<0.01）
- 変更履歴のRA1.2と比べると、Precision, Recall共に低下
 - StackOverflowのデータセットが多く、API使用パターンの組み合わせが多くなってしまったからだと考えられる

実験結果：組み合わせルール

	Precision	Recall
ベースライン	0.36	0.27
アプリ全部 (RA1.2)	0.75	0.22
回答全て (RA2.1)	0.66	0.13
RA1.2+RA2.1 (RA3)	0.77	0.21

- RA3はRA1.2よりもPrecisionがわずかに向上（p値=0.07）
 - StackOverflow手法のルールを組み合わせたため
 - StackOverflowのデータをより精査することでより精度が向上する可能性がある

まとめと所感

- まとめ

- 変更履歴・StackOverflowの回答に含まれるAPIに対して相関ルールマイニングを適用し、ベンチマークを作成
 - » StackOverflowからルールを作成したのは初！！
- StackOverflowなどの議論からAPI使用パターンを抽出することができる
 - » Precision向上, しかしRecallは低下

- 所感

- 「StackOverflowからルール作成」を思いついたのはすごい
- △ 開発者が実装したい機能を実現するAPIを推薦してくれるとより役立つと感じた
- △ StackOverflow手法を変更履歴で作ったテストデータで評価するのは少し違和感がある