Android アプリケーション開発支援ツールのための API 利用パターン情報収集における API バージョンの考慮の効果

原口 雄士, 近藤 悠志, 西本 匡志, 西山 佳志, 川端 英之, 弘中 哲夫

開発者にとって，オープンソースリポジトリマイニングによって得られる API 利用パターン情報は有用である．しかしながら，ライブラリやフレームワークによって提供される API は，バージョンアップによって新機能に関するクラスやメソッドが追加されたり，同じメソッド名でも仕様が変更されたりする．つまり API 利用パターン情報を収集する際に API のバージョンを正確に踏まえなければ，API メソッドの型や引数に関する情報が正確に得られない場合がある．結果として，収集される API 利用パターン情報が不正確になり，それを用いるアプリケーション開発支援ツールも影響を被る．これに対し我々は，収集された Android アプリケーションの API 利用パターン情報について，API のバージョン情報の踏まえる事による違いを調査した．その結果について報告を行う．

1 はじめに

近年のアプリケーションソフトウェアはライブラリやフレームワークの API (Application Programming Interface) を組み合わせることで作られている．さらに，オープンソースのライブラリやスマートフォン向けのフレームワークなどは，高頻度でアップデートされ続けている [2]．ライブラリやフレームワークのアップデートが行われると，API が追加あるいは削除されたり，既存の API に対する仕様変更が行れたりすることがある [1] [3]．ある調査によれば，Android の API は，API level が 16 から 26 になるまでの間，バージョンアップの度に平均して 57 個のクラスと 266 個のメソッドが追加，4 個のクラスと 10 個のメソッドが削除，155 個のクラスと 58 個のメソッドの仕様が変更されている [3]．

アプリケーション開発者には，実装内容に応じた適切な API の使い分けに関する知識や記述方法に関する知見が求められる．しかし，API の使用法や組み合わせ方に関する情報はドキュメント化されていないことも多い．さらにライブラリやフレームワークのバージョンによって使用可能な API には制約があり，開発者は API のバージョンを踏まえた上で使用する API を選択する必要がある．したがって，開発者が API の使用法に感する知見を得ることは容易ではない．

API の使用法を把握したい開発者にとって API メソッド呼び出しの順序系列や API メソッド集合などの，いわゆる API 利用パターン情報は有用な情報となりうる．実際，オープンソースリポジトリマイニングによって抽出された API 利用パターン情報を，API メソッドの推薦システム [7] [9] [10] やプログラムのバグ検出 [4] [5] などに活用する研究が数多くなされている．我々もプログラムのデータ依存関係を解析することで得られる API 利用パターン情報を活用したプログラム理解支援ツール [8] やコード自動編集機能付開発環境 [6] を開発している．

API 利用パターン情報を活用したアプリケーショ

Yushi Haraguchi, Yushi Kondoh, Masashi Nishimoto, Keiji Nishiyama, Hideyuki Kawabata, and Tetsuo Hironaka, 広島市立大学, Hiroshima City University.

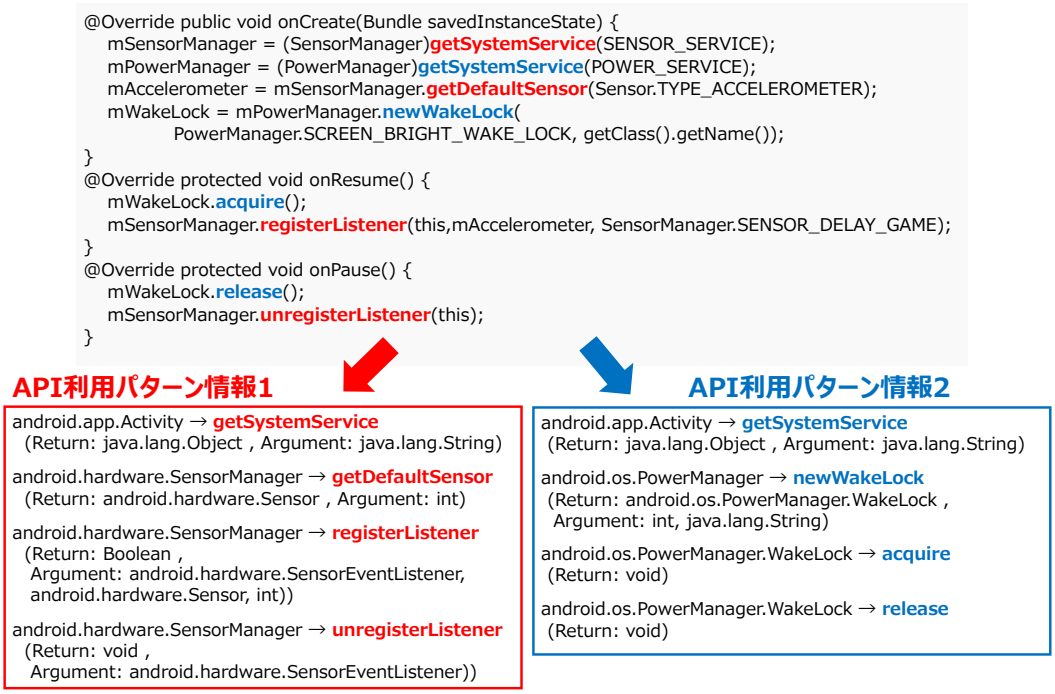

図 1　ソースコードから抽出される API 利用パターン情報群の例

ン開発支援ツールの有用性は，API 利用パターン情報の正確さや詳細さに大きく依存する可能性がある．正確かつ詳細な API 利用パターン情報を収集するための手段としては，API 利用パターン情報の収集する際に，プロジェクトに設定された API バージョン情報を正確に踏まえることが挙げられる．API 利用パターン情報を収集する際に，API のバージョン情報を正確に踏まえなければ，収集される API メソッドの型や引数に関する情報が不正確となる場合がある．また，API のバージョン情報が付与されていない API 利用パターン情報を使用するアプリケーション開発支援ツールは，開発者が使用する API バージョンには対応していない情報を提示するといった不正確な支援をしてしまう可能性がある．

本研究では，API 利用パターン情報を収集する際に，プロジェクトで設定されている．API のバージョン情報を正確に踏まえることが，API 利用パターン情報に与える影響について調査する．また，API のバージョン情報を正確に踏まえて収集した API 利用パターン情報がアプリケーション開発支援ツールに及ぼす影響について，我々が開発しているコード自動編集機能付開発環境 (以下，SSS) [6] を例に考察を行う．

2　準備

2.1　関連研究：API のバージョンの違いへの対応について

API のバージョンアップは，API を用いて構築されているソフトウェアの可用性に大きな影響を与えうる．API のバージョンアップによる影響を抑えるための研究として，代用可能な API を自動的に見つける方法に関する研究がある [3]．[3] では，Javadoc に頼ることの効果の少なさが指摘されるとともに，公式ドキュメントを踏まえて対策するのが確実であるとの報告がなされている．

多くの開発者は，アプリケーション開発にあたり，API のバージョンの違いを意識しようとする．実際

のところ，SDK_INT を用いて，複数のバージョンに対応した記述がなされるアプリケーションも多い．しかしながらその「複数バージョン対応」を正しく行うことは必ずしも容易ではない．文献[1] では，バージョンの違いを意識して記述されたアプリケーションプログラムが全体の 92%を占めることが報告されている．またその一方で，バージョンを意識しているいくつかのアプリのうち 25%にはバージョンの扱いに不適切な部分があったことも指摘している．

2.2 API 利用パターン情報への影響

図 1 は，我々が開発しているアプリケーション開発支援ツールで用いている API 利用パターン情報である [6] [8]．我々が活用する API 利用パターン情報は，特定の機能を実装するために必要とされる API メソッドの集合である．例えば，図 1 における API 利用パターン情報 1 はセンサー値を取得するために必要となる API メソッド集合であり，API 利用パターン情報 2 は WakeLock 機能を実装するために必要となる API メソッド集合である．抽出された各 API 利用パターン情報には，API メソッドの所属クラスとメソッド名，メソッドの戻り値の型と引数の型に関する情報が含まれている．なお，各 API メソッド名は，クラス名→メソッド名と本稿では表記される．

適切な API バージョン情報を踏まえてプログラムを解析しなかった場合，API メソッドが所属するクラスに関する情報や引数の仕様に関する情報に誤りが生じる場合がある．また，API のバージョンによって使用可能な API メソッドに制約がある場合もある．例えば，Android API の Progress Dialog クラスは，API Level 26 以降で非推奨となった API である．したがって，API のバージョンを 26 以上に設定している開発者に対して，Progress Dialog クラスを含んだ API 利用パターン情報を提示することは適切な支援であるとは言えない．

3 調査

本節では，オープンソースリポジトリマイニングによって API 利用パターン情報を収集する際に，API バージョン情報が正確か否かでどのような違いが見られるかを調査した．調査の手順としては，収集対象のプロジェクトに設定された API のバージョン情報を正確に踏まえて，抽出される API 利用パターン情報と，21 から 25 の API バージョンを設定して抽出される API 利用パターン情報の比較を行う．

3.1 調査の対象データ

本調査では，Google Samples[†1] における Android プロジェクトのうち，プロジェクトで使用されているライブラリが，Android 標準ライブラリと Android Support Library である 100 個のプロジェクトを調査対象とする．1 つの Android プロジェクトに複数のモジュールが含まれていた場合，各モジュールを 1 つの Android プロジェクトとして扱う．調査対象となる Android プロジェクトは，2017 年 1 月に収集したものを使用する．

表 1 各プロジェクトにおける API level

API level	Android プロジェクト数
23	11
24	89

3.2 調査内容

各 Android プロジェクトごとに正確な API バージョンで抽出した場合と，21 から 25 のバージョンを設定して抽出した場合の，API 利用パターン情報の違いを調査する．API 利用パターン情報は 3.1 から抽出を行う．API 利用パターン情報を自動抽出する際には，API 利用パターン情報抽出器に対して，プログラムの解析に使用する API のバージョン情報を与える必要がある．設定するバージョンは 21 から 25 の 5 つとした．サポートライブラリのバージョンにおいて指定する際には，21 の場合は *21.0.0*，25 の場合は *25.0.0* のようにした．API 利用パターン情報に含まれている各 API メソッドの情報として，API メソッドの名前の他に，引数や戻り値，所属するクラ

†1 https://github.com/googlesamples

スの型情報がある．今回は Android プロジェクトごとに正確なバージョンを踏まえて抽出した API 利用パターン情報と，21 から 25 のバージョンを設定して抽出した API 利用パターン情報を比較した時に，結果に違いのある API 利用パターン情報のみを扱う．

3.3 調査方法

はじめに，情報抽出元の各 Android プロジェクトと同じ API バージョンで API 利用パターン情報を抽出する．次に API バージョンを 21 から 25 に指定して，それぞれのバージョンで API 利用パターン情報を抽出する．ライブラリやフレームワークのバージョンを指定した場合は，抽出元の全ての Android プロジェクトに対し指定した API バージョンで API 利用パターン情報を抽出する．そして各 Android と同じライブラリやフレームワークで抽出した API 利用パターン情報と．それぞれ指定したバージョンで抽出された API 利用パターン情報について比較を行う．同じ API 利用パターン情報としての条件は，API 利用パターン情報を構成する各 API メソッドの情報が全て同じである事として比較を行う．

表 2　各バージョンごとにおける違いのある API 利用パターン情報の数

設定するバージョン	違いのあった API 利用パターン情報の数	抽出された API 利用パターン情報の総数
21	137	1817
22	129	1817
23	32	1865
24	2	1867
25	8	1867

3.4 結果

抽出元プロジェクトで使用されている API バージョンを，正確に踏まえて抽出した API 利用パターン情報と，21 から 25 のバージョンで抽出した API 利用パターン情報を比較し，違いのあった API 利用パターン情報の数を表 2 に示す．Android プロジェクトと同じライブラリやフレームワーク抽出した API 利用パターン情報の総数は 1866 であった．表 2 での設定するバージョンとは，ソースコード解析を行う際に用いるライブラリやフレームワークのバージョンである．それぞれの API 利用パターン情報を比較し，API メソッドの情報における違いとして特徴のある 3 つを挙げる．3 つの具体例を図 2 から図 3 に示す．

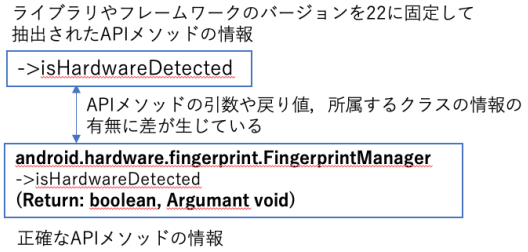

図 2　特徴 1

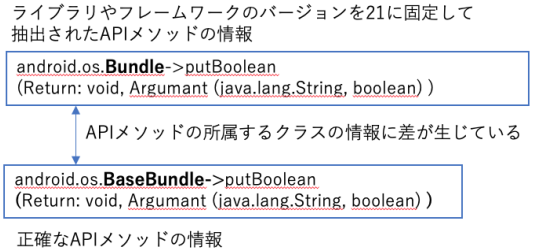

図 3　特徴 2

- （特徴 1 ）API メソッドの引数や戻り値に関する情報の有無．
- （特徴 2 ）API メソッドの引数や戻り値の情報は同じだが，所属するクラスの情報が異なる．
- （特徴 3 ）API メソッドのクラスは同じであるが，引数や戻り値の情報が異なる．

3.4.1 各パターンについて

それぞれの特徴 1 から特徴 3 の実例を図 2 から図 4 に示す．

特徴 1 では解析に用いたライブラリやフレームワークにクラス本体が定義されていなかったため，API メソッドの引数や戻り値，所属するクラスの情報が取

図 4　特徴 3

得できていない事が考えられる．

特徴 2 での違いのある 2 つのクラスは互いに継承関係のあるクラスであり，既存の API メソッドに対しオーバーライドされた API メソッドが存在している．図 3 の例において，”*putBoolean*” という API メソッドは API level が 22 のバージョンアップした際にオーバーライドされた API メソッドである．よって API level が 21 では ”*putBoolean*” という API メソッドはオーバーライドされておらず，API メソッドの所属するクラスの情報に違いが生じたのだと考えられる．

特徴 3 ではライブラリやフレームワークのバージョンアップにより，既存の API メソッドに対し仕様変更が行われたり，オーバーロードされた API メソッドであった．変更が行われる以前のバージョンでソースコード解析を行うと，変更後の API メソッドについての情報を参照する事ができず，正確な API メソッドの情報を取得できなかったと考えられる．

3.4.2　API 利用パターン情報への影響に関する考察

API 利用パターン情報を活用する開発者にとって，図 2 から図 4 のような不正確な情報は，有用な情報であるとは言えない．API バージョンを踏まえて API 利用パターン情報を抽出すると，そういった不正確な情報を正確に抽出する事ができた．

実際に正確かつ詳細な API 利用パターン情報が抽出できるようになった事で，我々が開発している API 利用パターン情報を活用するツールである，コード自動編集機能付開発環境 (以下，SSS) [6] を例に考察を行う．プログラムのデータ依存関係を解析することで収集した API 利用パターン情報を活用し，開発中のソースコードに開発者が選択したコード片を自動挿入するツールである．SSS によって挿入されるコード片の抽出元であるオープンソースプロジェクトが，開発者の使用する API のバージョンと異なっていた場合，開発中の環境ではコンパイルできないコード片が埋め込まれる可能性があった．しかし，API 利用パターン情報に API のバージョン情報が加味されたことによって，開発者が求める API バージョンのコード片を埋め込む事ができるようになると考えられる．

4　まとめと今後の課題

API 利用パターン情報を活用するツールにおけるアプリケーション開発者にとっての有用性は，API 利用パターン情報の正確さや詳細さに大きく依存する可能性がある．我々は，API のバージョンに着目し，正確かつ詳細な API 利用パターン情報の収集を行った．実際に今回の調査で，API のバージョン情報がない場合には API メソッドの情報について，不正確な情報が存在していた事がわかった．

今後の課題として，さらに API 利用パターン情報の正確さを向上させるために，API 利用パターンの抽出元プロジェクトを選定，もしくは収集した API 利用パターン情報の選別をする事で，開発者にとって有用性の高い API 利用パターン情報を収集できるのではないかと考える．また，API 利用パターン情報を活用したツールに関してより詳細な調査が必要である．

参 考 文 献

[1] He, D., Li, L., Wang, L., Zheng, H., Li, G., and Xue, J.: Understanding and Detecting Evolution-induced Compatibility Issues in Android Apps, *Proceedings of the 33rd ACM/IEEE International Conference on Automated Software Engineering*, ASE 2018, New York, NY, USA, ACM, 2018, pp. 167–177.

[2] Lamba, Y., Khattar, M., and Sureka, A.: Pravaaha: Mining Android Applications for Discovering API Call Usage Patterns and Trends, *ISEC*, 2014.

[3] Lamothe, M. and Shang, W.: Exploring the Use of Automated API Migrating Techniques in Practice: An Experience Report on Android, *Proceed-*

ings of the 15th International Conference on Mining Software Repositories, MSR '18, New York, NY, USA, ACM, 2018, pp. 503–514.

[4] 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, New York, NY, USA, ACM, 2005, pp. 306–315.

[5] Livshits, B. and Zimmermann, T.: DynaMine: finding common error patterns by mining software revision histories., Vol. 30, 01 2005, pp. 296–305.

[6] Masashi, N., Keiji, N., Hideyuki, K., and Tetsuo, H.: Easy-going Development of Event-Driven Applications by Iterating a Search-Select-Superpose Loop, *Journal of Information Processing*, Vol. 60, No. 3(2019).

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

[8] Nishimoto, M., Nishiyama, K., Kawabata, H., and Hironaka, T.: Supporting program understanding by automatic indexing of functionalities in source code, SERA 2019, May 2019, pp. 13–18.

[9] Xie, T. and Pei, J.: MAPO: Mining API Usages from Open Source Repositories, *Proceedings of the 2006 International Workshop on Mining Software Repositories*, MSR '06, New York, NY, USA, ACM, 2006, pp. 54–57.

[10] 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(2015), pp. 692–700.