Vol.2018-SE-200 No.10
2018/12/3

ソースコード変更履歴による Stack Overflow 記事の充実に向けて

西中 隆志郎[1,a)]　佐藤 亮介[1,b)]　亀井 靖高[1,c)]　鵜林 尚靖[1,d)]

概要：Stack Overflow（SO）は，API プログラミングに関する質問・回答投稿をコード片とともに共有しており，開発者の知識共有にしばしば用いられている．しかし，SO 投稿の量や回答時間は専門分野によって偏っており，近年，未回答の質問投稿の割合が増加する一因となっている．一方で，質問者と同じ問題に遭遇している開発者はどこかにいるはずであり，彼らが試行錯誤した経験はソフトウェアの開発履歴に蓄積されている可能性がある．本研究では，SO における知識共有の支援を目指し，ソースコード差分が SO を充実させる可能性を検討する．具体的には，特定の API に関するケーススタディとして，Android アプリケーションのリポジトリを対象にしてソースコード差分を取り出し，(1) Android SDK を含む何件の SO 内の投稿と関連付くか，(2) 時系列の観点から，ソースコードの差分が SO 内の投稿より早く取り出せるか，を調査した．その結果，Android SDK の種類名を用いて (1) SO 内の 24,184 件の投稿と関連付けることができ，(2) SO 投稿と関連付く差分 8,929 件のうち 29%が実際の SO 投稿よりも早くソースコードリポジトリの中から取り出せることがわかった．

キーワード：Stack Overflow，OSS，Android SDK，API，API usage pattern

1. はじめに

プログラミング言語やフレームワークの知識は，コーディングやデバッギング，またリファクタリングなどの複数の開発工程において重要である．開発者は，書籍や Web サイト，Wiki やディスカッションフォーラムなどから，開発に必要な知識を獲得する．特に Web サイトの情報は多くの人が手に入れやすく，作業中の開発者の知識獲得のためにしばしば利用される [2].

Web サイトの中でも, 特に Q&A サイトが利用される [18]. Q&A サイトはユーザからの質問（Question）側の投稿と，回答（Answer）側の投稿を記事として保存し，知識の共有を促す情報サイトである．質問側の投稿ではユーザの直面する問題が示されており，回答側の投稿では問題の解決方法が示されている．

Stack Overflow（SO）[*1] は登録ユーザ数が年々増加しており，開発者に有用な様々な情報を与える Q&A サイトとして期待されている [3]．SO はプログラミング技術に関する質問・回答に特化した Q&A サイトであり，コード片を投稿に含めて共有できる．例えば SO 投稿の中には特定の API の使用法に関する投稿記事が存在し，ユーザはコード片とともに API の使用法を調べることができる．公開されている API の中には，公式ドキュメントの内容が不足しているものも多く，そういったドキュメントの代替として SO の投稿が利用されている [12].

しかしながら，SO が得意とするプログラミング技術の領域は一部に限られている．SO 投稿の量は投稿の専門分野によってばらついており，特定の分野においては投稿が存在しない場合もある [12]．回答をもつ質問投稿に対して，未回答の質問投稿の割合は近年増加しており，その原因の 1 つとして，質問投稿の内容がユーザーの興味を引き付けない点や，質問内容がユーザの専門分野と異なるといった点が挙げられている [1]．また，質問投稿に回答が投稿されるまでの時間が，質問投稿の専門分野によりばらつく問題もある [10].

そこで本研究では，SO を充実させることにより開発者を支援することを目指す．その第一歩として，本稿では，ソースコード差分の内容が SO を充実させる可能性があるか否かを検討する．ソースコード差分に着目する理由は，情報の乏しい状況下でも，試行錯誤しながら問題に対処す

[1] 九州大学
Kyushu University
[a)] nishinaka@posl.ait.kyushu-u.ac.jp
[b)] sato@ait.kyushu-u.ac.jp
[c)] kamei@ait.kyushu-u.ac.jp
[d)] ubayashi@ait.kyushu-u.ac.jp
[*1] http://stackoverflow.com/

図 1　SO の質問投稿の事例

図 2　対応する回答投稿の事例

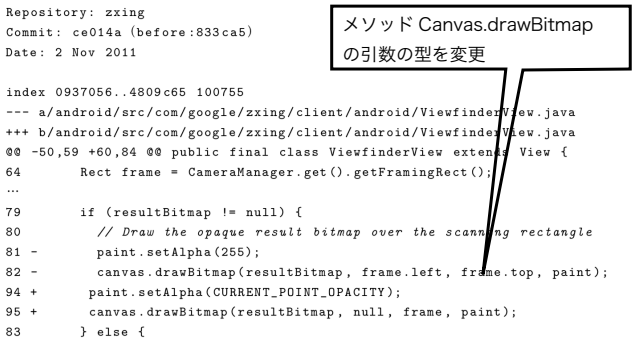

図 3　開発履歴中のソースコード差分の事例

る開発者は少なからず存在し，彼らの作業の情報はソフトウェアの開発履歴の中に存在し，抽出できる可能性があると考えたためである．またソースコード差分を機械的に生成することで，SO に質問や回答が投稿される時系列よりも早い段階で，関連する情報を開発者に提供できる可能性がある．本稿では，関連性の観点として API の種類の粒度に着目する．SO では，プログラミング言語やフレームワークといった API の粒度でしばしば議論が行われている [17]．

本稿では，ソースコード差分を生成するためのデータセットとして，Android アプリケーションリポジトリを用いる．Android SDK のドキュメントは一般的に不足していると指摘されており [11]，開発者が試行錯誤しながら開発を行なう可能性があると考えたためである．

差分の内容が SO 投稿を充実する可能性を検討するため，以下の 2 つの調査課題（RQ）に答える．

RQ1. ソースコード差分はどれだけの実際の SO 投稿に関連するのか　Q&A サイトは記事がなければ知識を共有できない．ソースコード差分が SO 記事を量的に支援できるかを第一に調べるため，差分と内容が近い SO 投稿の件数を計測する．具体的には，当該ソースコード差分の中に，SO 投稿で扱われている Android SDK メソッドが何件含まれているかで判断する．API のメソッド呼び出しは，プログラムの構造や開発者のタスクといったコンテキストを表すことが示唆されている [8]．Android アプリケーションソースコード内で使用される API にはいくつかの種類があるが，ドキュメントの不十分性が指摘されていることを鑑み，Android SDK を対象とする．

RQ2. ソースコード差分は実際の SO 投稿に対して早く生成できるか　SO の質問投稿の中には，回答が投稿されるまでの時間が極端に長いものが存在する．回答が投稿されるまでの時間的な課題を，ソースコード差分が改善できるかどうか探るため，SO 記事の投稿時刻に対するソースコード変更時刻の早さを調査する．RQ1 と同様に，ソースコード差分と SO 投稿との内容の近さも考慮に入れる．ソースコードの変更時刻は Git でコミットが行われた時間を用いる．

次章以降の構成は次のとおりである．まず 2 章では動機となる事例を挙げて本研究の動機を説明する．3 章では調査において用意するデータセットについて説明する．4 章では，調査のための調査課題（RQ）とその結果を詳細に述べる．5 章では関連研究について説明する．6 章で妥当性への脅威を述べ，7 章で結論を述べる．

2. 動機付けの事例

本稿では，実際の SO 記事を充実できるものとして，ソフトウェア開発履歴中のソースコード差分に着目する．本章では，本研究の目標とその動機となる事例について述べる．

図 1 と**図 2** は SO に投稿された質問投稿と回答投稿を示している．この投稿は，Android SDK の Canvas.drawBitmap メソッドの使用法を扱っている．この回答の元となった質問投稿では，Bitmap の動作が遅れるという問題が指摘されており，回答投稿では，その遅れを解消する方法として，drawBitmap メソッドの引数に渡すオブジェクトの初期化のタイミングの変更と，drawBitmap メソッドの引数の取り方の変更を教示している．

図 3 は，Android アプリケーション Barcode Scanner の開発履歴から取り出したソースコード差分を示している．コード内では，Canvas.drawBitmap メソッドの引数の内容が変更されている．具体的には第二引数が浮動小数点数型の frame.left から null に，第三引数が浮動小数点数型の frame.top から Rect 型の frame に変更されている．

Canvas.drawBitmap メソッドに関する変更の事例は，実際の SO 記事を充実させる 2 つの可能性をもつ．1 つ目は，

実際の SO 記事を量的に支援する可能性である．図 1 と図 2 のソースコード差分と関連する SO 記事は，いずれも Canvas.drawBitmap メソッドの引数の取り方に関するものであるため，内容的に関連している．ソースコード差分を生成すれば，Canvas.drawBitmap メソッドに関する問題と対処の情報を増加し，Canvas.drawBitmap メソッドの引数の取り方に興味をもつプログラマが関連する SO 記事を見つけやすくなる．仮に，Canvas.drawBitmap メソッドに関連する SO 投稿が多ければ，より多くの SO ユーザが，自身の問題に関連する情報を得やすくなる．

2 つ目は，SO における，問題や対処といった情報の蓄積を時系列的に早める可能性である．事例のソースコード差分を生成すれば，Canvas.drawBitmap メソッドの引数の取り方に興味をもつプログラマが，いち早く関連する問題と対処を見つけることができる．図 2 の SO 記事の回答の投稿時刻は 2014/02/11 であるが，ソースコードの変更時刻は 2011/11/02 であるため，ソースコード差分は，関連する実際の SO 記事よりも前の時刻に生成できることになる．

なお，この話題に関連する SO 記事は，2011/11/02 以前には存在しない．実際の SO において，android タグをもち，かつ Canvas.drawBitmap に関する記事のうち，2011/11/02 より以前に，drawBitmap の引数の取り方の変更を指摘した回答投稿は存在しなかった．

3. データセット

本章では，調査のために準備したソフトウェア開発履歴と，4 章で行う調査に用いる SO 投稿データの概要について述べる．

3.1 ソフトウェア開発履歴

ソフトウェアの入手：ソフトウェア開発履歴として F-Droid リポジトリ [5] を用いる．F-Droid は，Android に対応したオープンソースのアプリケーションパッケージを提供するアプリケーションストアであり，F-Droid の Web サイトでは，ソースコードを公開したページのリンクも提供している．F-Droid で公開されている 1,380 件（2017/12/19）のプロジェクトのうち，ソースコードが Git で管理されており，かつ Java ソースコードを持つ 713 件のプロジェクトを利用する．対象とするコミット数は 1,144,866 件である．

ソースコードの抽出：入手したリポジトリのソースコード履歴から，以下の 2 つの観点に沿ってソースコード差分を抽出する．

(1) バグ修正の行われた変更箇所
(2) Android SDK メソッドの呼び出しがある変更箇所

バグ修正の行われた変更箇所を特定するため，コミットメッセージを元にバグを修正したコミットを特定し，SZZ アルゴリズム [15] を用いてバグが混入したソースコードの箇所をクラスと行のレベルで特定する．SZZ アルゴリズムは，バージョン管理システムにより記録されたバグ報告メッセージとソースコードの差分の履歴から，バグを修正したコミットとバグが混入したコミットを特定する方法である．

本論文では，1）コミットメッセージに文字列 “fix bug” または “resolve bug” を含む，2）GitHub で管理された課題番号をコミットメッセージに含み，かつ該当する課題に “bug” ラベルが設定されている，のどちらかの条件を満たしたコミットをバグ修正が行われたコミットとした．次に特定したコミットで変更された行を遡り，バグが混入したとみられるコミットを特定した．

調査対象とする SDK：バグ修正の行われたコミットとバグが混入したコミットとの間で取った差分の中でも，Android SDK メソッドの呼び出しが含まれているものを対象とする．SO は API の使用法を共有・学習するためにしばしば使用されている [12]．特定のプロジェクトによって個別に実装されたインターフェースやクラス，およびメソッドは，SO での議論の範疇を超えていると考え，対象にしない．また Android SDK が継承する Java の標準ライブラリといった，Android SDK に属さないインターフェースやクラス，およびメソッドも対象にしない．標準的な Java や他の共通プラットフォームに関する API は，開発者の議論の対象となる場合が少ないという調査結果がある [7]．

Android SDK メソッド呼び出しの検出には正規表現を使用し，アプリケーションのパスが android で始まるメソッドのみを検出する．検出したメソッドが Android SDK であることを判断するため，Android Developer[*2][*3] で公開されているページをクローリングし，**表 1** に示す数の SDK を取得した．

表 1 対象とする Android SDK

プラットフォーム	パッケージ	クラス	メソッド
2（android，androidx）	320	3,777	183,068

表 2 データセットに用いる SO 投稿

投稿数（質問投稿）	投稿期間	条件
51,003	2008/08/07 〜 2017/03/13	「android」および「java」タグを質問投稿が所持

3.2 SO 投稿データ

調査対象とする SO 投稿の諸条件を**表 2** に示す．表 2 中の質問投稿に対応する回答投稿も調査の対象である．投稿期間は，Android SDK が公開される以前の投稿も含まれているが，投稿を「android」タグをもつものに限定することにより，Android SDK に関連するもののみを対象とする．

4. 調査と結果

生成したソースコード差分が実際の SO 記事を充実する可能性について調べるため，2 つの調査課題を設定し，調査

[*2] https://developer.android.com
[*3] 2018/08/01 時点

表 3 ソースコード差分で使用された Android SDK メソッドと差分の件数（上位 15 件）

Android SDK クラス (android プラットフォーム)	Android SDK メソッド	差分の数 (件)	クラス毎の合計 (件)
content.SharedPreferences	.getString	89	171
	.edit	82	
view.View	.findViewById	98	169
	.setVisibility	71	
view.LayoutInflater	.from	91	166
	.inflate	75	
content.ContentValues	.put	139	139
net.Uri	.parse	107	107
app.PendingIntent	.getActivity	98	98
view.Window	.setStatusBarColor	88	88
content.DialogInterface	.dismiss	86	86
app.Dialog	.dismiss	72	72
text.format.Time	.set	67	67
view.Menu	.findItem	66	66
app.NotificationManager	.notify	66	66

を行った．本章では，その方法と結果について議論する．

4.1 （RQ1）ソースコード差分はどれだけの実際の SO 記事に関連するのか

4.1.1 （RQ1.1）API の名前を介してどれだけの SO 記事に関連するのか

動機：ソースコード差分が実際の SO 記事を量的に支援する可能性を議論するため，差分と内容が近い実際の SO 記事の量を計測する．差分と内容が近い実際の SO 記事の量は，ソースコード差分が SO 記事に対して，関連する問題と対処を増加させる可能性であると考えた．

アプローチ：ソースコード差分と実際の SO 記事との内容の近さは，扱われた SDK メソッド名の列の一致で近似的に判断する．具体的な調査方法として，差分の中で使用されていた Android SDK メソッドのクラス名およびメソッド名が，投稿本文の文字列中に含まれる SO 記事を，差分と内容の近い SO 記事と判定する．

ソースコードの差分は Android アプリケーションのリポジトリ 713 件から抽出した 12,362 件のうち，2016/12/13 までに編集された差分 8,929 件を用いる．SO 投稿データは 2017/3/13 に投稿されたものまでであるため，この時刻の 3 ヶ月前より後に編集されたソースコードの差分は調査対象から除外する．実際の SO 記事には，3 章で説明した SO 投稿データを用いる．

ソースコード差分においてより多く扱われる Android SDK は，開発者が直面する本質的な問題と無関係である可能性がある．例えば，Log クラスのような標準的な SDK クラスは，異なる作業に共通し，かつ付随して使用される API であるため，開発者の作業とは直接的に関係しないものとして捉えられることがある [14]．そのため，より多くの差分で使用される下記の SDK メソッドは調査対象から除外している．

除外方法として，差分で扱われた全ての SDK メソッドの IDF（Inverse Document Frequency）値 idf を計算し，idf が特に小さく，かつ統計的に外れ値であったメソッドおよび Log クラスのメソッドを選択している．IDF は以下の計算式を用いる．

$$idf(t) = log(\frac{N}{df(t)}) + 1 \quad (1)$$

t は単一の SDK メソッドの名前である．$df(t)$ は SDK メソッドを含むソースコード差分の数であり，N は全ての差分の数である．本稿では $N = 8,929$ としている．対数 log の底は 2 としている．外れ値は，差分で扱われた全ての SDK メソッドの IDF 分布において，第 3 四分位点と 1.5 倍の四分位範囲の和より大きいもの，または第 1 四分位点と 1.5 倍の四分位範囲の差より小さいものとした．除外対象とした 9 件の SDK メソッドを以下に示す．

android.widget.Toast.makeText, android.content.Intent.putExtra, android.util.Log.d, android.util.Log.e, android.util.Log.i, android.util.Log.v, android.widget.TextView.setText, android.content.SharedPreferences.getBoolean, android.preference.PreferenceManager.getDefaultSharedPreferences

結果と考察：調査の結果，ソースコード差分と同じ Android SDK を含む SO 記事は 24,184 件であった．この数字は SO 投稿データセットの 47%を占める．また SO 記事と同じ Android SDK を使用するソースコード差分の件数は 7,692 件であった．

ソースコードおよび SO 投稿で扱われている SDK の具体的な種類について考察する．ソースコード差分において頻繁に使用される Android SDK クラスの上位 10 件および主要なメソッドの内訳を**表 3** に示す．

差分では，View クラスや LayoutInflater クラス，また PendingIntent クラスといった，UI に関する SDK がより多く使用されていた．こういった SDK を使用した差分は，UI に関する SO 上の質問に対応できる可能性がある．Joorabchi ら [6] によれば，開発者はモバイル開発において，GUI の検証に障害を抱えている．HCI ガイドラインがプラットフォームによって異なるために，GUI をテストすることが開発者にとって困難であることが示されている．

また実際の SO 投稿で頻繁に扱われる Android SDK クラスおよびメソッドの列の上位 10 件の分布を**表 4** および**表 5** に示す．SO 投稿においても，View クラスや Activity クラス，また Fragment クラスといった UI に関する SDK がより多く使用されていた．

差分で頻繁に使用される SDK と，SO 投稿で頻繁に扱われる SDK との間には，何らかの関連性があると考えられる．特に View クラスや Dialog クラスは差分で頻繁に使用されており，かつ SO 投稿でも頻繁に議論されている．使

表 4 実際の SO 記事で使用された Android SDK クラスと SO 記事の件数（上位 10 件）

Android SDK クラス	SO 記事数（件）
android.view.View	314
android.app.Activity	309
android.os.AsyncTask	157
android.app.Fragment	147
androidx.fragment.app.Fragment	142
android.widget.Button	98
android.widget.ListView	95
android.app.Dialog	87
android.text.format.Time	76
android.widget.ImageView	75

表 5 実際の SO 記事で使用された Android SDK メソッドと SO 記事の件数（上位 10 件）

Android SDK メソッド	SO 記事数（件）
android.view.View.findViewById	69
android.app.Dialog.show	65
android.widget.Toast.show	62
android.widget.Button.layout	62
android.app.Activity.onActivityResult	61
android.os.AsyncTask.doInBackground	60
android.widget.ListView.layout	60
android.view.View.post	50
androidx.fragment.app.Fragment.onCreate	49
android.app.Fragment.onCreate	49

表 6 除外対象の SDK メソッドに関するソースコード差分の変更と開発上の問題との関係性

Android SDK メソッド	差分の数 (件)	問題との関係性		
		関係あり	関係なし	判別不可
android.util.Log.d	22	0	21 (95%)	1 (5%)
android.util.Log.e	10	0	10 (100%)	0
android.util.Log.w	4	0	4 (100%)	0
android.widget.Toast.makeText	17	4 (24%)	3 (18%)	10 (59%)
android.content.Intent.putExtra	50	18 (36%)	5 (10%)	27 (54%)
android.widget.TextView.setText	7	0	1 (14%)	6 (86%)
android.content.SharedPreferences.getBoolean	3	0	0	3 (100%)
android.preference.PreferenceManager.getDefaultSharedPreferences	8	0	2 (25%)	6 (75%)

表 7 調査対象の SDK メソッドに関するソースコード差分の変更と開発上の問題との関係性

Android SDK メソッド	差分の数 (件)	質問との関係性		
		関係あり	関係なし	判別不可
android.support.v4.view.ViewPager.setOnPageChangeListener	2	2 (100%)	0	0
android.support.v4.widget.SwipeRefreshLayout.setRefreshing	2	1 (50%)	0	1 (50%)
android.view.View.setBackgroundColor	2	0	0	2 (100%)
android.graphics.Canvas.drawBitmap	1	1 (100%)	0	0

表 8 SO 投稿が含む Android SDK メソッドと実際の質問内容との関係性

Android SDK メソッド	SO 投稿の数 (件)	質問との関係性		
		関係あり	関係なし	判別不可
android.util.Log.i	193	6 (3%)	174 (90%)	13 (7%)
android.util.Log.e	104	0	94 (90%)	10 (10%)
android.util.Log.v	57	0	53 (93%)	4 (7%)
android.util.Log.d	31	1 (3%)	30 (97%)	0
android.util.Log.w	28	0	27 (96%)	1 (4%)

用される SDK の種類という粗い観点での可能性ではあるが，差分が SO 投稿の一部の専門分野に対応している可能性がある．

結果に対する妥当性を高めるため，除外対象の SDK メソッドのうち 8 件の SDK メソッドを使用した差分 121 件に対して，開発上の問題と関係するかを目視で確認した．この結果を**表 6** に示す．比較のために，調査対象の SDK メソッドのうち 4 件の SDK メソッドを使用した差分 7 件に対して，同様に目視で確認した結果を**表 7** に示す．

121 件の差分は，母数とした 8,929 件の差分のうち，無作為に抽出した 351 件の中で，除外対象の SDK メソッドを含んだものを選択した．確認した SDK が，除外対象の SDK9 件のうち 8 件である理由は，351 件の差分で使用されていたものが 8 件だったためである．

関係性の有無は，語句の部分的な一致により判断している．例えば，スワイプで画面を再更新した際に，機能が停止する問題があり，課題管理システムの議論文において，“swipe to refresh/board switch overlay bug” という報告がされていたとする．問題に対処したソースコードの変更において，問題の要因と見られる語句 “refresh” を名前に含むメソッド SwipeRefreshLayout.setRefreshing を呼び出した箇所が変更されていた場合 [*4]，当該メソッドの変更は問題と関係があると判断している．また別の例として，上述のような問題の詳細がコミットや課題の議論文に記載されている一方で，ソースコードの変更内容が，メソッド Log.d で出力する文字列を変更するといったものであった場合，関係がないと判断している．

さらに，除外対象の SDK メソッドのうち，SO 投稿内においても頻繁にみられる android.util.Log.d/e/w/i/v の 5 件の SDK メソッドを使用した SO 投稿延べ 413 件に対して，ユーザの質問と関係するかを目視で調査した結果を**表 8** に示す．

表 6，表 7 より，除外対象の SDK メソッドは，調査対象の SDK メソッドと比較して，開発上の問題との関係性が薄いと考えられる．除外対象の SDK において，開発上の

[*4] https://github.com/Luorrak/Ouroboros/commit/b496709415d4f9f0ee4b3213b（file:app/src/main/java/com/luorrak/ouroboros/catalog/CatalogFragment.java，line:184-194）

表 9 目視で確認したソースコード差分の内容（構文的な変更）

分類項目	差分の合計（件）
SDK メソッド呼び出しの削除	114
実引数自体の変更	80
リファクタリング（インスタンス名の変更，引数のインスタンス化）	27
SDK メソッド呼び出しの追加	17
インスタンスの型変更	7
異なる SDK クラスへの変更	6
SDK メソッドの引数の個数または型の変更	6
条件式の変更	6
インスタンス生成方法の変更	5
異なる SDK メソッドへの変更	5
SDK メソッド呼び出し順序の変更	3
例外処理分岐の追加	3

問題と関係する差分が存在した SDK は 8 件中 2 件だったのに対し，調査対象の SDK においては，同じ条件の SDK が 4 件中 3 件であった．また表 8 より，SO 投稿で使用された 5 件の SDK メソッドにおいて，差分において除外対象とした SDK と同程度の傾向がみられる．ユーザの質問と関係する投稿が存在した SDK は，5 件中 2 件であった．

ソースコード差分と同じ Android SDK を含む実際の SO 記事は，24,184 件であった．この数字は SO 投稿データセットの 47%を占める．Android SDK については，差分・SO 投稿ともに，UI に関するクラスがより多く扱われていた．

4.1.2 （RQ1.2）生成したソースコード差分はどういった問題や対処に関しているか

動機：ソースコード差分と実際の SO 記事の内容的な関連性をより具体的に探るため，ソースコード差分の内容を目視で調査する．

同じ SDK を使用したソースコード差分でも，変更の内容にはいくつかの種類と分類方法が存在する．単純な種類として，例えば SDK メソッドの引数の変更や，SDK メソッド呼び出し順序の変更といった種類がある．同様に，同じ SDK を扱った SO 記事の中でも，実装方法を議論した記事やメソッドの呼び出し順序の誤りを議論した記事といった種類がある．ソースコード差分と SO 記事を SDK の種類で関連付けるだけでは，こういった意味的な内容の一致を判別できない．

アプローチ：ソースコードの差分から無作為に抽出した差分 351 件の内容を目視で確認し，1）構文的な変更，2）変更を誘引した問題，の 2 つの観点で分類する．構文的な変更は，問題を発見するための手段において重要な役割を持つとされている [16]．また構文的な変更が同じ差分の間でも，開発者が直面していた問題は異なる可能性があるため，変更の元となった問題も調査する．Android アプリケーション開発における単純な問題として，例えば UI のバグやアプリビルドの不具合といったものが起こりうるが，そういった問題の内容を区別する．

表 10 目視で確認したソースコード差分の内容（変更を誘引した問題）

分類項目	差分の合計（件）
オブジェクト表示や画面操作，視覚効果といった UI の変更	29
ファイルアップロードや画像サイズ，早いスワイプによるクラッシュへの対処	9
データベース処理やストリーム接続，bluetooth 受信機といった機器に関する障害への対処	7
メモリリークや，描画する図形バッファのオーバーフローへの対処	3
Activity がキャッチできない例外やオブジェクトのタイトルがない例外への対処	2

無作為に抽出した 351 件の差分のうち 26 件は，変更された Android SDK が他の処理に付随して使用されるとみられるもののみであったため，その SDK 自体の問題による変更ではないとみなし，調査対象から除外した．除外した SDK は，android.util.Log クラスのメソッドおよび android.util.SparseArray クラスのメソッドである．

結果と考察：ソースコード差分の分類結果を**表 9**，**表 10** に示す．ソースコード差分が複数の項目に一致した場合は，各項目でソースコードの件数を加算している．表 9 中の項目「実引数自体の変更」として，文字列・数値型の値の変更と，他のメソッドを用いた引数の変換が該当する．項目「異なる SDK クラスへの変更」は，SDK の種類名が類似していた場合が該当する．例えば Android.view.MenuItem.getActionView メソッドの呼び出しを Android.view.MenuItemCompat.getActionView メソッドの呼び出しに変更したものだった場合がこの項目に該当する．

ソースコード差分の詳細な内容について考察する．表 9 より，最も頻繁に行われている構文的な変更は，SDK メソッド呼び出しの削除であった．一方で，SDK メソッドに渡される引数の型変更や，インスタンス生成方法の変更といった内容は比較的稀であった．こういった内容は，該当する差分は少ないものの，開発者によって頻繁に質問される項目であるため，SO ユーザの問題対処を支援する可能性がある．Duala-Ekoko ら [4] の調査によれば，開発者が作業中に抱く API に関する質問の中でも，特定のオブジェクトの生成方法や，特定のメソッドに渡す引数がもつ役割といった内容の質問がより多い．

また，表 10 の，差分の変更を誘引した問題の内容としては，UI やクラッシュ，また機器に関する問題が多かった．こういった内容も SO ユーザの問題対処を支援することが期待できる．Joorabchi ら [6] は，企業に所属する 9 人のモバイル開発者と 188 人のモバイル開発コミュニティに対してインタビューを行い，モバイル開発において，サポートされている解析ツールがプラットフォーム間で断片化されているために情報が不足し，開発者がクラッシュ解析に苦しんでいることを示している．UI やクラッシュ，また

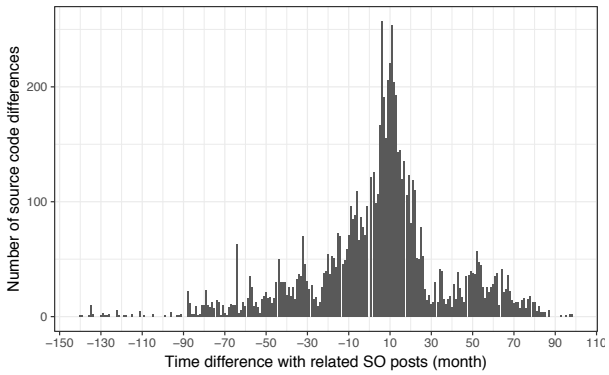

図 4 内容の近い SO 記事の投稿時刻との時間差間隔に対するソースコード差分の件数

機器に関する問題に対処した差分の内容は，こういった障害によって発見が遅れる問題を事前に減らすことが可能である．

構文的な変更内容として，SDK メソッド呼び出しの削除が頻繁に見られ，比較的稀なものに，SDK メソッドに渡される引数の型変更やインスタンス生成方法の変更といった内容が見られた．変更を誘引した問題として，UI やクラッシュ，また機器に関する問題が見られた．

4.2 （RQ2）ソースコード差分は実際の SO 投稿に対して早く生成できるか

動機：実際の SO 記事と内容が近いソースコード差分のうち，変更時刻が実際の SO 記事の投稿時刻よりも前だった場合，ソースコード差分は SO において投稿が不足している状態をいち早く改善できると考えられる．SO で質問が投稿されてから回答が投稿されるまでの時刻の中央値は 11 分だが，長い場合は 1 ヶ月を超える場合があるため，実際の SO 記事の代替となる SO 記事をいち早く生成できれば，SO における問題と対処の蓄積を時間的に早めることが期待できる [10][9].

アプローチ：実際の SO 記事に対してソースコード差分が早く生成できるかを調査するため，実際の SO 記事の投稿時刻に対するソースコードの変更時刻の差を計測する．ソースコードの変更時刻として，バージョン管理システムの Git でコミットが行われた時間を計測する．時間差の比較は，コミットが行われた時刻と SO 記事が投稿された時刻をそれぞれ Unix 時間に変換して比較する．

結果と考察：調査の結果を**図 4** に示す．図 4 の横軸は，SO の質問の投稿時刻に対する，内容が近いソースコードの変更時刻の差を月単位で示している．縦軸は，月単位の時間間隔毎のソースコード差分の件数を示している．ソースコード差分 8,929 件のうち，SO 記事と内容が近く，かつ SO 記事の投稿時刻より前に生成可能なソースコード差分は 2,203 件（29%）であった．短縮する時間として，実際の SO 記事に対して最大 140 か月前に生成でき，平均 29 か月前に生成できることがわかった．また興味深い点として，ソースコード差分の件数のピークは，内容が近い実際の SO 記事に対して約 9 か月後である．すなわち，特定の SDK に関して，実際の SO で議論されてから約 9 か月後に，開発現場でより頻繁に使用されていることがわかる．

ソースコード差分は，SO 記事の投稿時刻より前に生成可能である．差分 8,929 件のうち，SO 記事と内容が近く，かつ SO 記事の投稿時刻より前に生成可能なソースコード差分は 2,203 件（29%）であった．

5. 関連研究

知識を共有する媒体としてプログラミング Q&A サイトの情報に注目し，API ドキュメントを補う可能性の調査が行われている [12]．しかし Asaduzzaman らによると，Q&A サイトの情報もまた完全ではなく，不足した領域が存在していると報告されている [1]．SO では回答のない質問投稿の割合が年々増加しており，その要因の一つとして，SO ユーザが投稿に付与するタグの不正確さが挙げられている．SO ユーザは，質問投稿に付与されたタグを用いて，技術的な領域と実際の質問投稿を紐づけている．回答のない質問投稿の中にはタグが正確でない投稿が存在しており，タグ付けの不正確さが，知識を持った SO ユーザを引き付けるのに失敗する要因の一つとして考えられている [1].

知識を持った SO ユーザが質問投稿にたどり着くのを助けるため，Smrithi ら [13] は，質問を投稿しようとするユーザへの適切なタグの自動推薦に取り組んでいる．彼らは，多項式ナイーブベイズと SVM の 2 つの分類子を利用し，投稿に添付されたコード片からコードの記述に使用されたプログラミング言語を判別し，タグを生成している．手法を評価するため，50,000 件の質問投稿を用意し，75%を訓練データ，残りを検証用データとしてタグの生成精度を測定したところ，72%の精度で正しいタグを付与している．また特定の質問に関連する全てのタグと，最も関連性の高いタグの 50%を識別する結果が得られている．

一方本稿では，不足する SO 投稿のより広範囲な領域に着目し，ソースコード差分を利用することで実際の SO 投稿の充実を目指している．回答のない質問投稿として，タグが不正確である質問投稿の他に，高度に専門的な投稿や，コード片が添付されていない投稿といった投稿が存在する [1]．本稿において，こういった SO 投稿も充実の対象としている点が既存の研究と異なる．

6. 妥当性への脅威

RQ1.1 の調査環境における脅威として，ソースコード差分の分類結果が，調査員の知識や技術に依存する点が挙げられる．差分の分類は，ソースコードの内容が特定の開発

状況に依存するか否かを本稿の筆頭著者が単独で判断したものであり，その判断基準は筆者の知識に依存する．そのため筆者の知識量が分類結果に影響を及ぼす可能性がある．

RQ1 の実験結果に関する脅威を挙げる．除外対象とした Android SDK の中には，ノイズではない SDK が存在する可能性がある．ノイズでない SDK として，特に SDK メソッド android.content.Intent.putExtra が挙げられる．

本稿では，突出して多くのソースコード差分で使用される SDK メソッドは，開発上の問題と無関係であると想定し，調査対象から除外した．しかし，多くの差分で使用され，かつ開発上の問題と関係する SDK も存在する可能性がある．Wang ら [19] によれば，Intent クラスは，Android アプリケーションリポジトリにおいて頻繁に使用されており，かつ SO 投稿のタグに設定される場合が多い．すなわち，開発上で頻繁に使用されており，かつ開発者の議論の対象となる場合も多いクラスであることが示唆されている．

7. まとめと今後

本稿では，開発者が API を使用したコード片を手掛かりとし，Android アプリケーションリポジトリから生成したソースコード差分が現実の SO 記事を充実する可能性を調査した．今後の研究では，開発者が残したソースコード以外の情報についても，存在する SO 記事と関連付くか調査する予定である．ソフトウェア開発履歴中にはソースコードだけでなく，コミットメッセージや課題管理システムに残された議論の詳細といった自然言語情報も存在する．またソースコード差分が，SO に存在していない質問と回答を充実する可能性の調査も今後の課題である．回答のない質問投稿を補う取り組みとして，ソースコード差分から擬似的な SO 記事を機械生成するプロトタイプツールを筆者は作成している [20]．擬似的な SO 記事が，実際の SO の質問と回答に代わる可能性も調査する予定である．

謝辞 本研究は，JP26240007，18H04097 による助成を受けた．

参考文献

[1] Asaduzzaman, M., Mashiyat, A. S., Roy, C. K. and Schneider, K. A.: Answering Questions About Unanswered Questions of Stack Overflow, *Proceedings of the 10th Working Conference on Mining Software Repositories*, Piscataway, NJ, USA, IEEE Press, pp. 97–100 (2013).

[2] Brandt, J., Guo, P. J., Lewenstein, J., Dontcheva, M. and Klemmer, S. R.: Two studies of opportunistic programming: interleaving web foraging, learning, and writing code, *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, ACM, pp. 1589–1598 (2009).

[3] Chen, F. and Kim, S.: Crowd debugging, *Proceedings of the 2015 10th Joint Meeting on Foundations of Software Engineering*, ACM, pp. 320–332 (2015).

[4] Duala-Ekoko, E. and Robillard, M. P.: Asking and answering questions about unfamiliar APIs: An exploratory study, *Software Engineering (ICSE), 2012 34th International Conference on*, IEEE, pp. 266–276 (2012).

[5] F-Droid: Free and open source android app repository, [Online; accessed 2015-12-19] (2017).

[6] Joorabchi, M. E., Mesbah, A. and Kruchten, P.: Real challenges in mobile app development, *Empirical Software Engineering and Measurement, 2013 ACM/IEEE International Symposium on*, IEEE, pp. 15–24 (2013).

[7] Kavaler, D., Posnett, D., Gibler, C., Chen, H., Devanbu, P. and Filkov, V.: Using and asking: Apis used in the android market and asked about in stackoverflow, *International Conference on Social Informatics*, Springer, pp. 405–418 (2013).

[8] Kim, J., Lee, S., Hwang, S.-W. and Kim, S.: Enriching documents with examples: A corpus mining approach, *ACM Transactions on Information Systems (TOIS)*, Vol. 31, No. 1, p. 1 (2013).

[9] Lai, L.-C. and Kao, H.-Y.: Question routing by modeling user expertise and activity in cQA services, *The 26th Annual Conference of the Japanese Society for Artificial Intelligence* (2012).

[10] Mamykina, L., Manoim, B., Mittal, M., Hripcsak, G. and Hartmann, B.: Design lessons from the fastest q&a site in the west, *Proceedings of the SIGCHI conference on Human factors in computing systems*, ACM, pp. 2857–2866 (2011).

[11] Nguyen, T. T., Pham, H. V., Vu, P. M. and Nguyen, T. T.: Learning API Usages from Bytecode: A Statistical Approach, *Proceedings of the 38th International Conference on Software Engineering*, New York, NY, USA, ACM, pp. 416–427 (2016).

[12] Parnin, C., Treude, C., Grammel, L. and Storey, M.-A.: Crowd documentation: Exploring the coverage and the dynamics of API discussions on Stack Overflow, Technical report, Georgia Institute of Technology (2012).

[13] Rekha, V. S., Divya, N. and Bagavathi, P. S.: A hybrid auto-tagging system for stackoverflow forum questions, *Proceedings of the 2014 International Conference on Interdisciplinary Advances in Applied Computing*, ACM, p. 56 (2014).

[14] Saied, M. A., Benomar, O., Abdeen, H. and Sahraoui, H.: Mining multi-level api usage patterns, *Software Analysis, Evolution and Reengineering (SANER), 2015 IEEE 22nd International Conference on*, IEEE, pp. 23–32 (2015).

[15] Śliwerski, J., Zimmermann, T. and Zeller, A.: When do changes induce fixes?, *ACM sigsoft software engineering notes*, Vol. 30, No. 4, pp. 1–5 (2005).

[16] Sun, B., Shu, G., Podgurski, A. and Robinson, B.: Extending static analysis by mining project-specific rules, *Proceedings of the 34th International Conference on Software Engineering*, IEEE Press, pp. 1054–1063 (2012).

[17] Treude, C., Barzilay, O. and Storey, M.-A.: How Do Programmers Ask and Answer Questions on the Web?, *Proceedings of the 33rd International Conference on Software Engineering*, New York, NY, USA, ACM, pp. 804–807 (2011).

[18] Vasilescu, B., Serebrenik, A., Devanbu, P. and Filkov, V.: How social Q&A sites are changing knowledge sharing in open source software communities, *Proceedings of the 17th ACM conference on Computer supported cooperative work & social computing*, ACM, pp. 342–354 (2014).

[19] Wang, W. and Godfrey, M. W.: Detecting api usage obstacles: A study of ios and android developer questions, *Proceedings of the 10th Working Conference on Mining Software Repositories*, IEEE Press, pp. 61–64 (2013).

[20] 西中隆志郎，鵜林尚靖，亀井靖高，佐藤亮介：ソフトウェア開発履歴情報からの API Q&A 知識の自動抽出，第 24 回ソフトウェア工学の基礎ワークショップ，FOSE, pp. 147–152 (2017).