特別研究報告

題目

Stack Overflow に投稿された Java API を含むコード片の有効性の調査

指導教員
井上 克郎 教授

報告者
西村 広太郎

令和 2 年 2 月 10 日

大阪大学 基礎工学部 情報科学科

令和元年度 特別研究報告

Stack Overflow に投稿された Java API を含むコード片の有効性の調査

西村 広太郎

内容梗概

Stack Overflow とはプログラマのための Q&A サイトであり，質問者は質問にコード片を付与することでより具体的な質問内容を投稿することができる．さらに，投稿にはタイトルやタグといった情報を追加することができ，閲覧者は容易に興味のある情報にたどり着くことができる．また，投稿者ではないユーザが投稿内容の編集を提案することが可能であり，編集によって投稿内容が改善されると判断された場合，編集が適用される．

一方，Stack Overflow には API を利用する際に生じた問題に関連する質問が数多く投稿されている．これらの投稿は API を利用するユーザだけでなく，API の開発者にとっても有益な情報となるため，API の利用に関連する投稿を対象とした研究が行われている．しかし，ライブラリは日々更新されており，Stack Overflow の投稿のうち，過去に投稿されたものに含まれる API が現在も使用可能であるとは限らない．

そこで本研究では，Java で利用される 5 種類のライブラリに含まれる API を対象に，Stack Overflow 上の投稿に含まれるコード片の API がどの程度現在も使用可能かを調査する．具体的には，投稿内容からコード片を抽出し，利用されている API を検出する．検出された API がライブラリの最新バージョンで使用可能かどうかによって，コード片に含まれる API が現在も使用可能か判定する．コード片に含まれる全ての API が使用可能であればコード片は有効であるとし，使用が非推奨になっているか削除されている API を含んでいれば有効でないとする．

この調査手法によって，各ライブラリに含まれる API を利用したコード片について，情報の有効性を判定することができた．また，有効でないと判定したコード片について，API の使用が非推奨になった時期が，ライブラリのあるバージョンアップの時期に多く集中していることを確認した．

主な用語

Java API
Stack Overflow
SOTorrent

目 次

1　まえがき

API は，開発者がよく利用する機能を外部のプログラムから呼び出せるものとして，ソフトウェア開発において頻繁に利用される．Java では，API をメソッドで実装しており，実装されたプログラムファイルをまとめたものをライブラリと呼ぶ．

近年，開発者のための Q&A サイトでの交流が活発に行われている．そのような Q&A サイトの一つに，Stack Overflow[5] があり，多数の質問や回答が投稿されている．Stack Overflow には API を利用する際に生じた問題に関する投稿が数多く投稿されている．API の利用者は，このような投稿を閲覧することで API の利用方法や利用の際の注意事項を確認する．また，API の開発者もこれらの投稿を参考にしてユーザがどのように API を利用しているのか，API にどのような機能を求めているのかを確認する．このため，API の利用に関連する投稿は関心を引きやすく，多くの研究の対象となっている．

しかし，それらの投稿中に含まれる API の利用方法がライブラリの最新バージョンでも正しいものであるとは限らない．Java で利用されるライブラリにおいて，ライブラリのバージョンアップの際に使用が非推奨となった，あるいは削除されることにより使用できなくなった API を使用した投稿が存在した場合，その投稿に含まれる情報は古くなっていると判断できる．

本研究では，投稿に含まれる全ての API がライブラリの最新バージョンで使用可能である投稿を有効な投稿，使用できない API を含む投稿を有効でない投稿と定義する．この定義に基づき，Stack Overflow 上の投稿に含まれるコード片の API がどの程度ライブラリの最新バージョンで使用可能であるかを調査し，その有効性を判定した．また，有効でないと判定された API を含む投稿については，いつからその API が有効でなくなったかを調査した．なお，本研究では Java で書かれたコード片を含む投稿を調査対象とし，関連する投稿が Stack Overflow に多数見つかった 5 種類のライブラリについて調査を行った．

以降，2 章では研究背景について述べる．続いて，3 章では調査対象や調査手法について述べ，4 章では調査結果について述べる．5 章でまとめと今後の課題について述べる．

2　研究背景

2.1　API

API（Application Programming Interface）とは，ソフトウェアの機能を外部のプログラムから呼び出して利用するための手順や形式を定めた規約である．ユーザは開発の際に API を利用することで機能を実装する手間を省くことができる．API は，プログラム言語ごとに実装方法が異なる．本研究において，API とは Java によって実装されている API を指すものとする．Java において，API はクラスファイルの中でメソッドの形で実装される．クラスファイルはライブラリとしてまとめられ公開される．また，API の開発者は公開したライブラリを更新することがある．更新内容は様々あり，公開した API をよりユーザが使用しやすいように改善するといった更新や，API に欠陥が発見されたためライブラリから削除するといった更新がある．

API がユーザにとって便利なものである一方，API の使用について理解することは難しい [7][8]．この理由の一つとして，API とドキュメントの間で整合性が保たれない可能性が挙げられる．API 開発者は，公開した API についてその使用方法を記述したドキュメントを合わせて公開することが一般的だが，ライブラリの更新で API の内容が変更された際にドキュメントの更新が行われず，ドキュメントと API の間で内容の乖離が起こることがある．これによってユーザが API の使用方法を理解することが困難になることがある．

2.2　Stack Overflow

Stack Overflow とは，開発者向けに開設された Q&A サイトである．質問者は投稿の際に，タイトルやタグといった情報を追加することで質問内容が何に関連するものであるかを閲覧者に提示することができる．さらに，質問内容に関連する URL や，記述したソースコードの一部ないしは全部を本文に貼り付けることができ，内容を詳細に記述することが可能である．また，Stack Overflow には信用度と呼ばれるスコア制度が存在する．全てのユーザはアカウントを作成した際に 1 の信用度を与えられ，この値は表 1 に示す条件を満たすことで変動する．また，回答の報酬として信用度を提示することもでき，この場合は質問者が提示した値分信用度が変動する．信用度が上がることでユーザには権限が付与され，投稿にプラス投票やマイナス投票をすることが可能となる．2000 以上の信用度を持つユーザは信頼に値するとされ，投稿者でなくても投稿の編集をすることが可能となる.[1]編集は，質問や回答の文章が読みにくい，内容に不備があるなど投稿に改善の余地があると感じた場合に行われる．信用度の低いユーザでも編集をするべき投稿と編集内容を提案することができるが，編

[1]https://stackoverflow.com/help/privileges

集内容を適用するには信頼されたユーザからの承認が必要となる．この編集作業によってサイト全体の投稿の質が向上し，閲覧者にとって読みやすいサイトが作られている．

表 1: 信用度

信用度の変動条件	変化量
質問がプラス投票される	+10
回答がプラス投票される	+10
回答が承認される	+15
回答を承認する	+2
編集内容が承認される	+2
質問がマイナス投票される	-2
回答がマイナス投票される	-2
誰かの回答にマイナス投票する	-1
自分の投稿が 6 回通報される	-100

また，Stack Overflow には API に関連する問題について，多数の質問が投稿される．これは，ユーザにとって API について理解することが困難であるということを示している．API に関連する投稿は同じような問題に直面しているユーザにとって重要なものであるが，多数の投稿の中から自分の知りたい情報に関連する投稿を見つけることは難しい．この問題を解決するため，Stack Overflow を対象にした研究で，API に関連する投稿について取り扱うものは数多く存在する．

Ahasanuzzaman ら [4] は，API に関連する投稿は API 開発者にとっても，ユーザが API にどのようなことを要求しているのかを知ることができるという点で高い価値を持つと考えた．そこで，投稿のうち API に関連するものを検出する手法として，教師あり機械学習の一種である CAPS という手法を提案した．

Subramanian ら [9] は，API を使用しているコード片と API の使用方法を記した公式ドキュメントとのリンクを生成する手法を提案し，その手法を用いたツールとして Baker を実装した．Subramanian らは，開発した Baker がコード片に含まれる API を適切に識別できるか評価するため，Stack Overflow の投稿データを利用した．Baker の評価について，この研究では precision の値を重視している．Baker は Java と JavaScript の 2 言語に対応しており，Java では 0.98，JavaScript では 0.97 といずれも高い precision の数値を記録し，Baker の有用性を評価した．

Zhou ら [10] の研究では，26 のオープンソースな Java ライブラリとフレームワークを対象に API の非推奨化と削除の履歴を解析した．全体の 40%以上の非推奨 API は削除されてい

るが，システムによってばらつきがあり，半分のシステムは非推奨 API を一度も削除しておらず，15.4%のシステムだけが非推奨 API を削除していた．また，非推奨の API や削除された API は，ほぼすべてがバージョン 1.0 に到達る前かメジャーリリース間での変更で起きており，それ以外で削除されたものは数個にとどまった．また，Zhou らは，Android API について，Stack Overflow 上のコードが非推奨であるかどうかを検出するツール Deprecation Watcher を作成した．このツールを評価するために，タグに Android をもつ投稿の承認された回答を対象に，回答に含まれるコード片で使用が非推奨であるものを検出したところ，precision の値は 1.0，recall の値は 0.86 と高い精度で API を検出できることが確認できた．

2.3 SOTorrent

SOTorrent[2] は，Stack Overflow が公開している XML データダンプを利用して作成されたデータセットである．XML データダンプはこれまで Stack Overflow に投稿された質問，回答やユーザ情報といったデータを投稿単位でまとめたものである．

SOTorrent は，Stack Overflow 解析用のデータセットで，編集によって日々更新されていく Stack Overflow の投稿を解析するという目的のもと，Baltes らによって作成された．その内容は Stack Overflow が公開している XML ダンプのデータを更に詳細にしたものとなっており，投稿内容の編集前後の情報を属性として持つ，テキストブロックとコードブロックを識別できる，といった特徴がある．文献 [2] から，SOTorrent のデータベーススキーマの全容を図 1 に引用して示す．

また，SOTorrent を用いた研究は数多く行われている．Rahman ら [6] の研究では，Stack Overflow 上の投稿のうち，Python に関連する投稿について，どれだけの頻度でセキュアでない回答が出現するかを調査した．この研究で SOTorrent は，コード片の抽出や承認された回答を見分けるための手段として利用されている．

Abric ら [1] の研究では，Stack Overflow 上の重複した質問について，その影響を調査した．Abric らは SOTorrent を利用して，ユーザの過去の信用度やアクティビティを知ることで重複する質問をするユーザの性質の調査を可能とした．

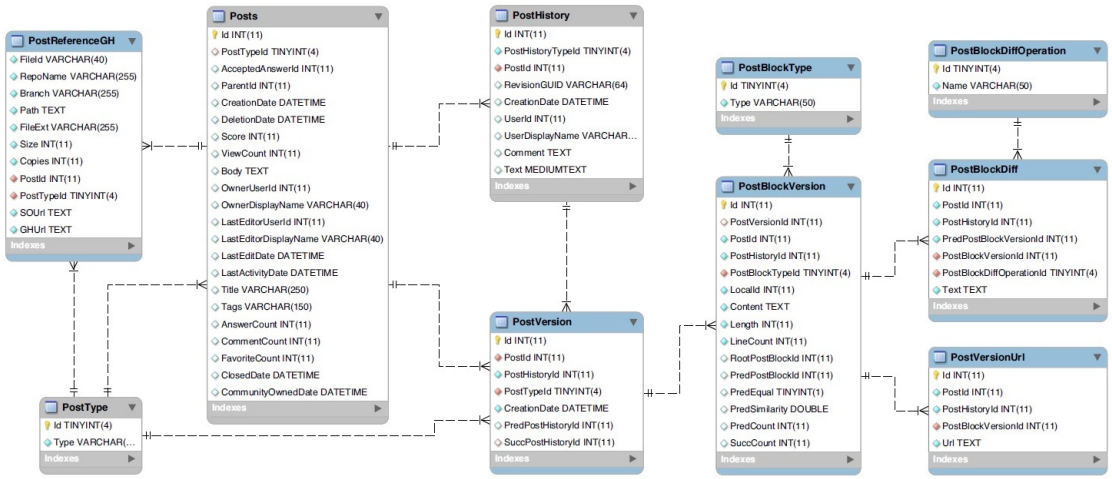

図 1: SOTorrent データベーススキーマ

3 調査手法

3.1 調査目的

近年，Stack Overflow に API に関する質問が数多く投稿されている．特に，API に関する質問は数多く投稿され，ユーザにとって API を理解することは困難であるということを示している．しかし，Stack Overflow に投稿されて長い年月が経過した質問や回答については，API が現在使用できない可能性がある．理由として，ソフトウェア開発に利用されるライブラリの情報が日々更新されるため，過去に投稿された情報が古くなることが挙げられる．

本研究の目的は，Stack Overflow 上の投稿について投稿中のコード片に含まれる API がライブラリの最新バージョンでも使用可能であるかどうかを調査し，実際に Stack Overflow 上の情報がどの程度使用できなくなったのかを明らかにすることである．次節では，API が使用可能であるかどうかを判定するためにどのような観点から調査を行うかについて述べる．

3.2 調査項目

本研究では，Stack Overflow に投稿された質問や回答に含まれるコード片で使用されている API がライブラリの最新バージョンに含まれているかどうかを調査する．バージョンに

ついては 2020 年 1 月時点でリリースされているものを最新のバージョンとする．また，ライブラリの最新バージョンに含まれているかどうかについて，本研究ではメソッド名と引数の観点から判断する．コード片とライブラリの最新バージョンそれぞれに含まれる API 呼び出しについてメソッド名と引数の数と型が一致している場合，コード片中の API が最新バージョンに含まれていると判断し，API の情報が有効であるとみなす．これまでのバージョンでは使用可能であったものの，最新バージョンにおいて使用が非推奨になっているものや削除されているものは有効でないとみなす．コード片に含まれる全ての API について情報が有効である場合はコード片が有効であるとし，情報が有効でない API を含むコード片についてはコード片が有効でないとする．

また，有効でないと判断されたコード片については，いつから有効でなくなったのかを調査する．具体的には，コード片に含まれるライブラリの最新バージョンで有効でない API がどのバージョンから有効でなくなったのかを調査する．

3.3 調査対象

本研究では，Stack Overflow に投稿された質問や回答に含まれるコード片のうち，Java で書かれたコード片について調査する．コード片に含まれる API については，Java のクラスライブラリに含まれるものを対象に調査する．以下に，調査対象とするコード片とライブラリについて述べる．

3.3.1 対象コード片

調査対象とする Java のコード片は，以下の条件全てを満たす投稿に含まれるコード片である．

- タグに Java を含むもの
- タイトル，本文，タグのいずれかに調査対象のライブラリ名を含むもの
- 投稿データが最新であるもの

これらの条件の設定理由について以下に述べる．はじめに，Stack Overflow では，160 万件を超える投稿がタグに Java を持つ．Java で書かれたコード片を分類するために，タグに Java を含む投稿に含まれるコード片を調査の対象とする．次に，調査対象のライブラリに関連するコード片を選択するために，調査対象のライブラリ名をタグに持つものを条件に設定する．ただし，Stack Overflow では信用度が 1500 以上のユーザでなければ新しくタグを作成することができない．そのため，ライブラリ名がタグとして作成されていない時期に投稿されたものを調査対象外としないために，タグだけではなくタイトル，本文にライブラリ

名が含まれる投稿もライブラリに関連するものである可能性が高いため調査の対象とする．最後に，コード片は編集作業によって内容が投稿時点のものと現在のものとの間で異なる可能性がある．本研究では現在の API について有効性を調査するため，現在ユーザが閲覧できる編集後の投稿データを対象とする．

3.3.2 対象ライブラリ

今回調査対象とするライブラリについて，表 2 に詳しい情報を示す．なお，各ライブラリの最古バージョンは，GitHub リポジトリに存在する最も古いリリースのバージョンとし，最新バージョンは，2020 年 1 月時点で最新のリリースのものとする．

表 2: 調査対象ライブラリ

ライブラリ名	最古バージョン		最新バージョン	
Guava	Ver2.0	(2010/01/12)	Ver28.2	(2019/12/27)
JavaMail	Ver1.4.1	(2007/11/13)	Ver1.6.2	(2018/08/30)
Jsoup	Ver0.1.1	(2010/01/31)	Ver1.12.1	(2019/05/13)
SLF4J	Ver1.0RC4	(2009/08/21)	Ver1.7.30	(2019/12/17)
Twitter4J	Ver2.0.4	(2009/05/19)	Ver4.0.7	(2018/08/16)

表 2 に挙げたライブラリは Java を用いた開発においてよく使用されており，これらに関連する質問は Stack Overflow に 1000 件以上と数多く存在する．また，ライブラリバージョンを調査するにあたり，GitHub 上にリポジトリを持つという基準でこれらのライブラリを調査対象とした．

3.4 調査手順

3.4.1 手順 1：Java で書かれたコード片の抽出

投稿データから Java で書かれたコード片を選択するにあたり，はじめに SOTorrent データセットを用いたデータベースの構築を行う．現在，SOTorrent の最新バージョンでは，2019 年 12 月 2 日更新分までのデータが使用可能となっている．本研究では，2019 年 6 月 21 日更新分までのデータセットを調査対象として使用した．

本研究では，図 1 のテーブルのうち，Posts，PostBlockVersion，PostBlockType に保存されているデータを利用する．表 3，4，5 に各テーブルのカラムのうち本研究で利用するものとその説明を示す．

表 3: Posts

カラム名	説明	データ型
Id	投稿 ID	Integer
PostTypeId	投稿のタイプを識別するための ID	TinyInt
CreationDate	投稿日時	DateTime
Body	投稿本文	Text
Title	投稿タイトル	VarChar(250)
Tags	投稿タグ	VarChar(150)

表 4: PostBlockVersion

カラム名	説明	データ型
Id	投稿のバージョン ID	Integer
PostBlockTypeId	投稿のタイプを表す ID	TinyInt
PostId	投稿バージョンの元となる投稿 ID	Integer
Content	投稿内容	Text
MostRecentVersion	編集後最新のものかどうか	Boolean

表 5: PostBlockType

カラム名	説明	データ型
Id	投稿タイプを表す ID	TinyInt
Name	投稿タイプの名前	VarChar(50)

データベース構築後，前処理としてデータベースから Java で書かれたコード片を抽出し，抽出したコード片を含む投稿の投稿日時，タイトル，投稿本文，コード片の内容，投稿に付けられたタグ，投稿が最新バージョンかどうか，という情報を新しいテーブルに保存した．以下にその手順を述べる．

まず，Posts からタグに Java を含むデータの ID，投稿日時，本文，タイトル，タグの情報を取得する．

次に，PostBlockVersion からコード片，投稿データが最新のものであるかどうかの情報を取得する．このとき，PostId の値が先ほど Posts から取得した Id の値と一致するもので，投稿のタイプがコードブロックであるものを指定する．これにより，投稿に含まれるコード片を対象にデータを取得できる．

データベースから取得したこれらのデータは，JavaPost という名前で新しく作成したテーブルに保存する．表 6 に JavaPost の詳細な説明を示す．この先の手順で調査対象のライブラリに関連するコード片を選択する際は，JavaPost から対象コードを選択する．

表 6: JavaPost

カラム名	説明	データ型
Id	データを一意に識別するための ID	Integer
PostId	投稿を識別するための ID	Integer
CreationDate	投稿日時	DateTime
Title	タイトル	VarChar(250)
Body	投稿本文	Text
CodeBody	コードブロック内のコード片	Text
Tags	投稿に付けられたタグ	VarChar(150)
MostRecentVersion	投稿が最新バージョンかどうか 最新バージョンであれば true	Boolean

3.4.2 手順 2：ライブラリデータベース構築

続いて，対象となるライブラリに含まれる API 情報をデータベースに登録する．本研究では，GitHub にリポジトリが存在するライブラリを取り扱う．はじめに，GitHub からライブラリのリポジトリをクローンする．

クローンしたリポジトリから，API が実装されている java ファイルを探索し，ライブラリのバージョン，リリース日，そのバージョンに含まれる API のメソッド名と引数の数，型

を API 情報としてデータベースに登録する．また，このとき API が非推奨なものかどうかを同時に登録する．

3.4.3　手順 3：対象コード片の解析

続いて，手順 1 で作成したテーブルから対象のライブラリに関連するコード片を選択する．

JavaPost からタイトル，本文，タグのいずれかにライブラリ名を含むもので，データが最新バージョンであるものを選択し，コード片と投稿日時を取得する．

次に，取得したコード片を解析してコード片に含まれる API を抽出する．Java では，API はメソッドの形で実装されているため，メソッド名と引数の数と型を抽出する．コード片の解析には JavaParser[3] を用いた．

このとき，解析不可能なコード片も数多く存在した．主な原因として，クラス宣言の欠落や図 2 に例として示す三点リーダーのようにコード中に現れる省略表現による解析エラーが挙げられる．これらは投稿内のコードブロックが長くなってしまう際に質問内容と直接関係のない冗長な部分を省略し，投稿を読みやすくするという理由によるものだと考えられる．これらの解析不能なコード片のうち，クラス宣言が欠落しているコード片については，図 3 に示すようにクラス宣言部分を補うことで解析を可能とした．コード中の省略表現など，クラス宣言部分の補完のみでは解析が不可能であると判断されたコード片については，本研究では対象外とした．

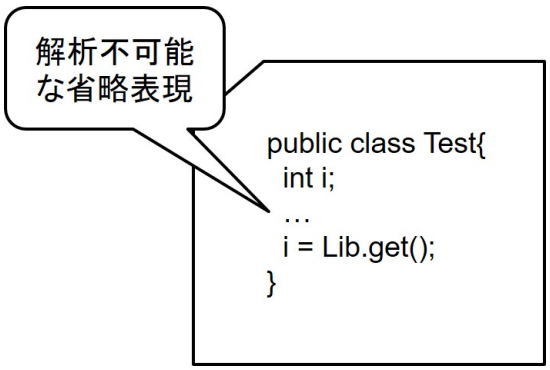

図 2: コード片中に見られる省略表現の例

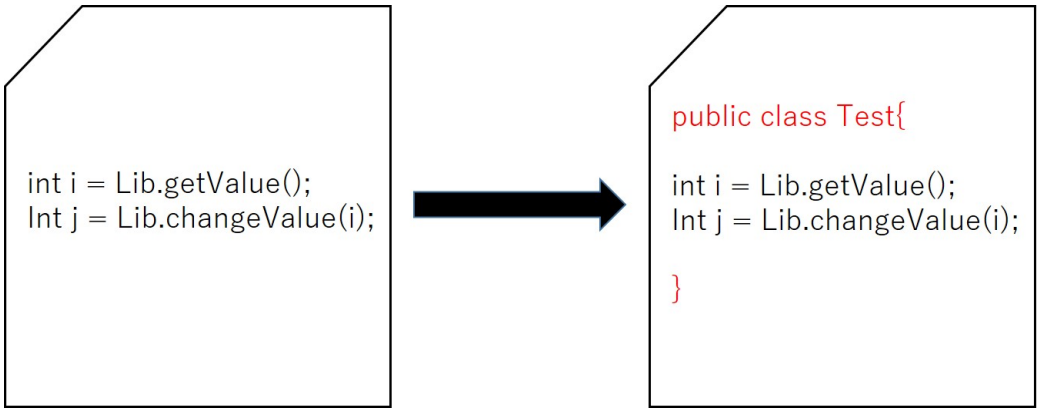

図 3: クラス宣言部分を付与した例

3.4.4 手順 4：API の照合

抽出したコード片に含まれる API をライブラリデータベースに登録されている情報と比較する．

手順 3 で抽出した API から，メソッド名と引数の数と型を参照し，ライブラリデータベースに登録されている API のメソッド名と引数の数と型を比較する．一致するものが発見された場合，API を利用していると判断する．利用されている API について，ライブラリのバージョンごとに API が使用可能かどうかを判定する．あるバージョンで，コード片に含まれる API が全て使用可能であれば，そのバージョンにおいてコード片は有効であると判定し，非推奨か削除された API が含まれていれば，有効でないと判定する．

ライブラリの最新バージョンでコード片が有効であれば，コード片は現在も有効であるといえる．コード片に最新バージョンで有効でない API が含まれている場合，API が有効でない最も古いバージョンのリリース日をコード片が有効でなくなった日付とする．

図 4 は，あるコード片について，最新バージョンが Ver2.0 のあるライブラリに含まれる API を照合した結果の例を示す．コード片には 3 種類の API が含まれており，あるバージョンで有効であれば○，有効でなければ×を記入している．図 4 より，ライブラリの最新バージョンで有効でない API を含んでいるため，コード片は有効でないと判定する．また，過去のバージョンにおける有効性について，API が有効でない最古のバージョンが Ver1.1 であるため，このコード片は Ver1.1 のリリース日から有効性を失ったと判定する．

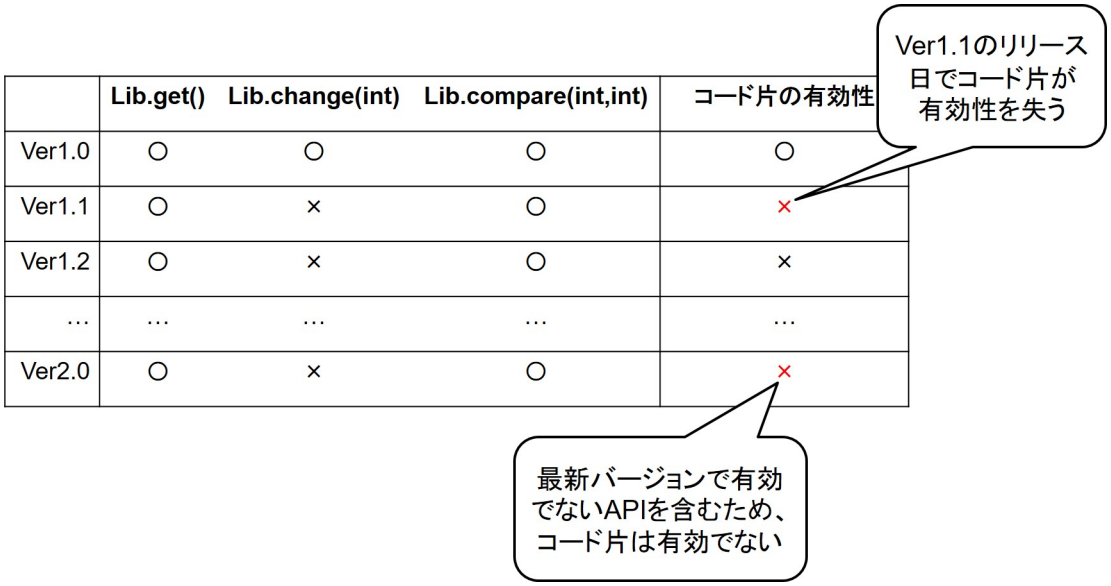

図 4: API の照合結果の例

4　調査結果

本章では，前章で述べた調査対象のコード片に含まれる API 情報について，その調査結果を述べる．まずはじめに，調査手順の各手順において解析できたコード片の数を報告し，その後 3.2 節で述べたそれぞれの調査項目について結果を示す．

4.1　調査手法で解析したコード片の数

Stack Overflow のコード片を解析するにあたり，様々な理由で本手法で有効かどうか判定できないものがある．ここでは，調査手法で述べた各手順ごとにどれだけのコード片が解析可能であったかを示す．

4.1.1　各ライブラリに関連するコード片の数

表 7 は，手順 1 で抽出したコード片のうち，どれだけが手順 3 で解析可能であったかを示したものである．表から，今回の調査手法で解析可能なコード片の数はライブラリに関連するコード片の約 20〜30%にとどまっていることが確認できる．原因として，3.4.3 節で述べた省略表現によって解析不可能なコード片が多く存在することが挙げられる．

表 7: 手順 3 における解析可能なコード片の数

ライブラリ名	各ライブラリに関連したコード片の数	解析可能であったコード片の数
Guava	11032	2857
JavaMail	8863	2178
Jsoup	10990	2847
SLF4J	25025	5616
Twitter4J	1603	437

表 8: 手順 4 における API 利用が確認できたコード片の数

ライブラリ名	解析可能であったコード片の数	API を含むコード片の数
Guava	2857	808
JavaMail	2178	1156
Jsoup	2847	1843
SLF4J	5616	1366
Twitter4J	437	215

4.1.2 API 利用が確認できたコード片の数

表 8 は，手順 3 で解析可能であったコード片のうち，どれだけが手順 4 で API の有効性判定を行えたかをしめしたものである．ライブラリに関連するコード片が実際にライブラリに含まれる API を含んでいるとは限らないため，有効性の判定をすることができるコード片の数が減少している．API を含むコード片に対して，調査項目で示した各項目について調査を行った．

4.2 項目 1：各ライブラリにおけるコード片の有効性

次に，各ライブラリに関連するコード片の有効性について，調査結果を表 9 に示す．

表 9 において，有効な API のみを含むコード片数は，コード片に含まれる API がすべてライブラリの最新バージョンで使用可能であるものを示す．有効でない API を含むコード片数は，コード片中に，最新バージョンで使用できない API を含むものを示す．また，有効であったコード片の割合とは，API を含むコード片の数のうち有効な API のみを含むコード片の数の割合を示す．表 9 から，Guava を除く 4 種類のライブラリについて，関連する

コード片はすべてライブラリの最新バージョンでも有効性を保っていることが確認できる．また，Guava に関連するコード片についても，95%を超えるものが有効であることが確認できた．

この調査結果について，Guava は，他の 4 種類のライブラリと比較して頻繁にバージョンアップが行われる．そのため，非推奨になる API や削除される API が他のライブラリと比較して多く存在しており，そのような API を含むコード片が Stack Overflow 上に残っていると考えられる．

4.3 項目 2：有効でないコード片の生存期間

前節で述べた Guava に関連する投稿のうち情報が有効でない 31 のコード片について，投稿日から情報が有効でなくなるまでの期間を調査した．図 5 は，その結果をグラフにしたものである．グラフの横軸は時間を表す．各コード片について，投稿された日時を要素の左端として，情報が有効であった期間を青色で表し，情報が有効でない期間を赤色で表す．コード片に含まれる有効でない API を目視で確認し，その API が非推奨になったあるいは削除されたライブラリバージョンのリリース日をコード片が有効でなくなった日と定義した．

図 5 より，2017 年直後において，投稿に含まれる API が有効性を失っていることが確認できる．この結果について，2017 年 5 月に行われた Guava のバージョンアップでライブラリの内容が大幅に変更されたため，使用が非推奨になった API が増加したことを確認した．そのため，Guava に関連するコード片を含む投稿のうち，2017 年 5 月以前に投稿されたものについては，コード片に含まれる API が有効でない可能性があるため注意が必要であるといえる．

表 9: 各ライブラリにおけるコード片の有効性

ライブラリ名	API を含むコード片の数	有効な API のみを含むコード片の数	有効でない API を含むコード片の数	有効であったコード片の割合
Guava	808	777	31	0.96
JavaMail	1156	1156	0	1.0
Jsoup	1843	1843	0	1.0
SLF4J	1366	1366	0	1.0
Twitter4J	215	215	0	1.0

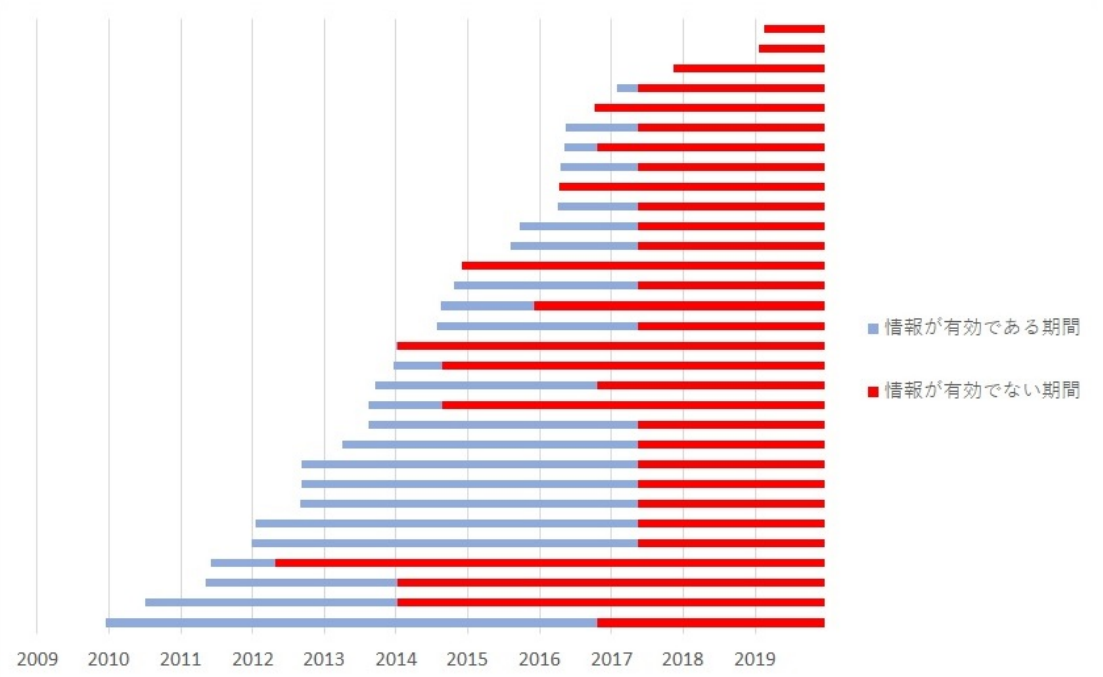

図 5: 投稿日時から API が有効性を失うまでの期間

5　まとめと今後の課題

本研究では，Stack Overflow 上の投稿に含まれるコード片について，コード片に含まれる API が最新のライブラリバージョンで使用可能かどうかを調査した．調査は Java の 5 種類のライブラリに含まれる API を対象に行い，また，投稿に後から編集が加えられているものについては最新のコード片を対象に行った．

調査結果では，4 種類のライブラリについて調査したコード片に含まれる API はすべて有効であることを確認した．また，有効でない情報が発見された 1 種類のライブラリについても多数のコード片が有効であり，有効でないコード片の多くは 2017 年直後に有効性を失っていることを確認した．

今後の課題を挙げる．1 つ目の課題は，解析が不可能であったコード片に含まれる API 情報の調査である．今回の研究では，JavaPerser でコード片の解析を行い，解析不可能なコード片については調査対象外とした．これらのコード片についても有効性を調査することで，Stack Overflow 上の投稿に含まれるコード片についてより詳細な結果を得ることができると考えられる．

2 つ目の課題は，他言語への調査対象の拡張である．Stack Overflow には Java に関する質問だけでなく，JavaScript や Python といった他言語に関連する質問も数多く投稿されて

いる．Stack Overflow の投稿に含まれるコード片で使用される，これらの言語で利用される API についても調査が必要である．

謝辞

大阪大学大学院情報科学研究科コンピュータサイエンス専攻 井上 克郎 教授には，研究活動及び研究室での生活において貴重な御意見及び御指導を賜りました．井上 教授に厚く御礼申し上げます．

大阪大学大学院情報科学研究科コンピュータサイエンス専攻 松下 誠 准教授には発表における問題点の提示において，多くの御助言を賜りました．松下 准教授に心より深く感謝いたします．

大阪大学大学院情報科学研究科コンピュータサイエンス専攻 神田 哲也 助教には研究活動における直接の御指導及び本論文執筆における御助言など，多くの場面で御支援を賜りました．神田 哲也 助教の御支援のおかげで本論文の完成に至りました．神田 助教に心より深く感謝いたします．

大阪大学大学院情報科学研究科コンピュータサイエンス専攻 伊藤 薫 氏，嶋利 一真 氏，小笠原 康貴 氏，溝内 剛 氏，白木 秀弥 氏，原口 公輔 氏には，本論文執筆や，発表における問題点の提示など，日々研究に関する多くの御助言を賜りました．先輩諸氏に心より深く感謝いたします．

最後に，その他様々な御指導，御助言等を頂いた大阪大学大学院情報科学研究科コンピュータサイエンス専攻井上研究室の皆様に心より深く感謝いたします．

参考文献

[1] D Abric, O E. Clark, M Caminiti, K Gallaba, and S McIntosh. Can Duplicate Questions on Stack Overflow Benefit the Software Development Community? In *Proc. of the International Conference on Mining Software Repositories (MSR)*, p. To appear, 2019.

[2] S. Baltes, L. Dumani, C. Treude, and S. Diehl. Sotorrent: reconstructing and analyzing the evolution of stack overflow posts. *Proceedings of the 15th International Conference on Mining Software Repositories*, pp. 319–330, 2018.

[3] JavaParser. https://javaparser.org/, (Accessed on 02/10/2020).

[4] Ahasanuzzaman M, Asaduzzaman M, Roy CK, and Schneider KA. Classifying stack overflow posts on api issues. *international conference on software analysis, evolution and reengineering*, p. 244–254, 2018.

[5] Stack Overflow. https://stackoverflow.com/, (Accessed on 02/10/2020).

[6] A Rahman, E Farhana, and N Imtiaz. Snakes in paradise?: Insecure python-related coding practices in stack overflow. *2019 IEEE/ACM 16th International Conference on Mining Software Repositories (MSR)*, pp. 200–204, 2019.

[7] M. P. Robillard. What makes apis hard to learn? answers from developers. *IEEE Software*, Vol. 26, No. 6, pp. 27–34, 2009.

[8] M. P. Robillard and R. DeLine. A field study of api learning obstacles. *Empirical Software Engineering*, Vol. 16, No. 6, pp. 703–732, 2011.

[9] S. Subramanian, L. Inozemtseva, and R. Holmes. Live api documentation. *Proceedings of the International Conference Software Engineering*, pp. 643–652, 2014.

[10] J. Zhou and R. J. Walker. Api deprecation: A retrospective analysis and detection method for code examples on the web. *FSE 2016: Proceedings of the 2016 24th ACM SIGSOFT International Symposium on Foundations of Software Engineering*, p. 266–277, 2016.