Vol.2021-SE-208 No.13
2021/7/9

GitHubプロジェクトに利用されている Stack Overflowのコード片の進化パターンの調査

栗原 拓己[1,a)]　嶋利 一真[1,b)]　神田 哲也[1,c)]　井上 克郎[1,d)]

概要：ソフトウェア開発プラットフォームである GitHub では，プログラミングに特化した Q&A フォーラムである Stack Overflow からコード片の再利用が行われている．Stack Overflow のコード片は頻繁に変更が行われ，利用者はその修正状況を把握することが望ましい．しかし，Stack Overflow のコード片と再利用先の GitHub のコード片はそれぞれ異なった変更が繰り返されており，2 つのプラットフォームで同様の変更がされているものはごくわずかであるという報告がなされている．そのため，Stack Overflow のコード片の進化を追従する必要性についてはより詳細な分析が必要である．本研究では，GitHub プロジェクトに利用されている Stack Overflow のコード片の変更履歴を分析し，進化パターンの分類を行う．対象は Java プロジェクトに限定し，データセット SOTorrnet を用いて，再利用された Stack Overflow のコード片の変更履歴を手作業で調査した．GitHub のコード片が StackOverflow のコード片に追従しているか調査し，追従していないものに対しては進化パターンの分類を行い，進化を追従する必要性について考察した．

1. はじめに

ソフトウェア開発において，ソースコードの再利用は頻繁に行われている．特にソフトウェア開発プラットフォームである GitHub では，さまざまな Web サイトからコード片の再利用が行われている．その中でも，Stack Overflow のようなプログラミングに特化した Q&A フォーラムは盛んに利用されており，開発者はコーディングに関する質問をし，回答を求めることで効率よく開発を行っている．Stack Overflow では質問と回答が公開されているため，質問者以外の開発者も投稿に含まれるコード片を再利用することが多い．Hata らの調査によると，GitHub プロジェクトのソースコード中にコメントとして記述されるドメインについて，Stack Overflow は 2 番目に多く参照されていることが明らかになっており，開発者が頻繁に Stack Overflow の投稿を参照していることが分かる [1]．しかし，Stack Overflow の投稿は修正などの目的で変更が加えられることが多く，投稿の再利用先で最新バージョンでない投稿が利用され続けている可能性がある．Manes らの調査では，GitHub プロジェクトで参照されているコード片を含む Stack Overflow の投稿のうち 79%が，少なくとも一度は変更されていることが明らかになっている [2]．しかし，Manes らの異なる調査で，参照関係にある GitHub プロジェクトと Stack Overflow のコード片において，ともに進化しているものは高々 10%程度で，それ以外は独立に進化していることが明らかになっている [3]．この調査では両者のコード片の変更内容について詳細な分析が行われておらず，バグ修正などの再利用時に影響を及ぼすような Stack Overflow のコード片の変更を，コード片を利用した GitHub プロジェクトの開発者が見逃している可能性が存在する．

そこで本調査では Java プロジェクトを対象として，GitHub プロジェクトに利用されている Stack Overflow のコード片の進化に関する調査を行い，進化のパターンについて分析を行う．また，これらのコード片の内容や GitHub プロジェクトにおける追従状況に関しても調査し，Stack Overflow のコード片を再利用した際に，その後のコード片の進化に追従する必要性について考察する．本研究の RQ は以下の通りである．

RQ1: Stack Overflow の最新バージョンのコード片を利用している GitHub プロジェクトはどの程度あるか？

GitHub プロジェクトにおける Stack Overflow のコード片の利用事例に関して，最新バージョンの投稿が利用されている割合に関して調査を行う．調査可能な 232 件中 186

[1] 大阪大学
[a)] takumi-k@ist.osaka-u.ac.jp
[b)] k-simari@ist.osaka-u.ac.jp
[c)] t-kanda@ist.osaka-u.ac.jp
[d)] inoue@ist.osaka-u.ac.jp

件が最新バージョンのコード片を利用しており，46 件が旧バージョンのコード片を利用していた．

RQ2: GitHub プロジェクトに旧バージョンが利用されている Stack Overflow のコード片はどのような進化パターンがあるか？

GitHub プロジェクトが利用した Stack Overflow のコード片と現在の Stack Overflow のコード片に差分がある場合に，ユーザから見た振舞いの変化をもとにコード片の進化パターンを分類した．その結果，旧バージョンを利用している 46 件中振舞いの変化があるものが 17 件，振舞いの変化がないものが 25 件，Stack Overflow の投稿の説明を補足したものが 4 件であった．また，それぞれの進化パターンに対して進化を追従する必要性があるか考察した．

以降，2 章では研究背景や関連研究について述べる．3 章では本研究の調査方法について述べ，4 章では，得られた結果と考察を述べる．最後に 5 章では本研究のまとめと今後の課題を述べる．

2. 背景

2.1 Stack Overflow

Stack Overflow は 2008 年に設立されたプログラミングに関する質問とそれに対する回答を投稿・検索できる Q&A フォーラムである [4]．2021 年 6 月現在，ユーザ数は約 1,500 万人，投稿は質問と回答を合わせて約 5,300 万件で，1 日に約 1,000 万人が Stack Overflow を閲覧しており，開発者に広く普及している [5]．開発者は，機能の実装方法や API の利用法などに関する疑問が生じた際に，その内容をテキストやコード片を組み合わせて説明し質問として投稿する．質問に対する回答も同様にテキストやコード片を組み合わせて投稿でき，質問や回答に関するコメントや投稿内容の変更も可能である．また，利用者は質問や回答の変更履歴も閲覧可能である．図 1 に回答の変更履歴の例を示す．回答例では 3 つのバージョンがあり，回答に対する二度目の編集では実線で囲まれたコードブロックにおいて，赤でハイライトされたコードが削除され，緑でハイライトされたコードが追加されていることが分かる．これらの編集は回答者だけでなくの他のユーザが変更可能であり，寄せられたコメントに関連した内容の変更も多い [6]．

2.2 SOTorrent

SOTorrent[6] は，公式の SO データダンプと Google BigQuery GitHub データセットに基づくオープンデータセットであり，投稿のテキストブロックやコードブロックごとの変更履歴が保存されている．図 2 にテキストブロックとコードブロックに分けた Stack Overflow の回答例を示す．図 2 で破線で囲まれている番号 1,3 のブロックがテキストブロック，実線で囲まれている番号 2 はコードブロックである．番号 4 はこの回答に寄せられたコメントである．SOTorrent を用いることでそれぞれのブロックごとのバージョン履歴にアクセスでき，詳細な分析が可能である．また，SOTorrent には Stack Overflow の投稿を参照した GitHub プロジェクトの情報も格納されている．この情報を用いることで，GitHub プロジェクトにおける Stack Overflow のコード片の利用状況を調査することができる．

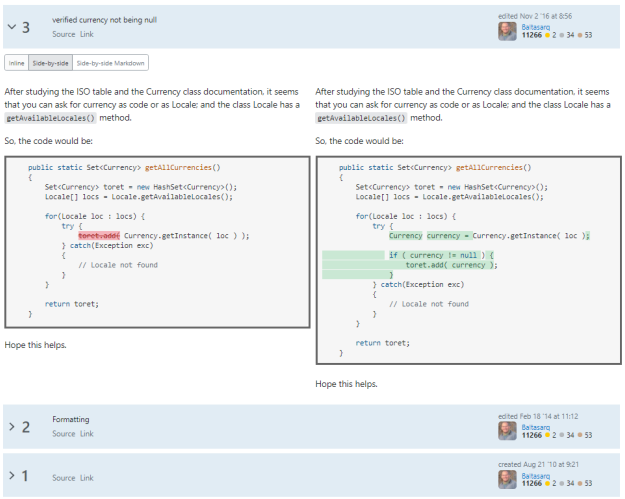

図 1 Stack Overflow の投稿の編集履歴（ID:3537085）[*1]

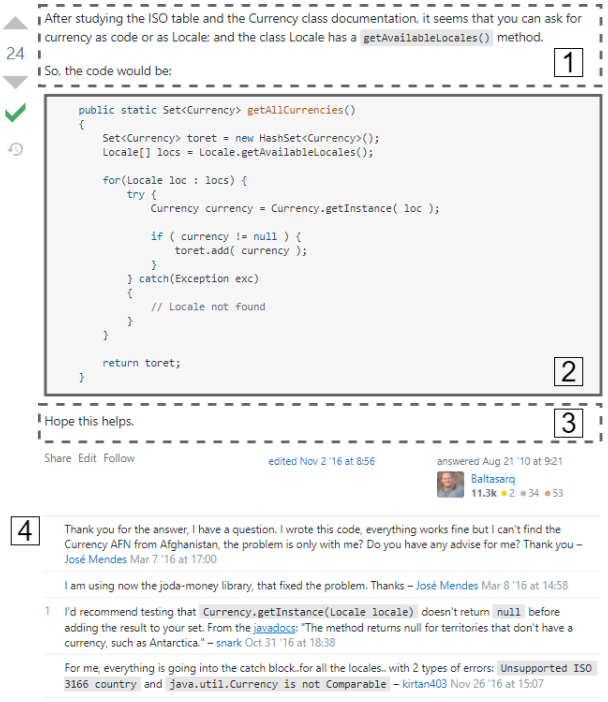

図 2 ブロックごとに分けられた Stack Overflow の回答例（ID:3537085）[*2]

2.3 Stack Overflow の投稿の再利用

ソフトウェア開発における機能の実装の際に，すでに存在

*1 https://stackoverflow.com/posts/3537085/revisions

*2 https://stackoverflow.com/questions/3536968/get-all-possible-available-currencies/3537085#3537085

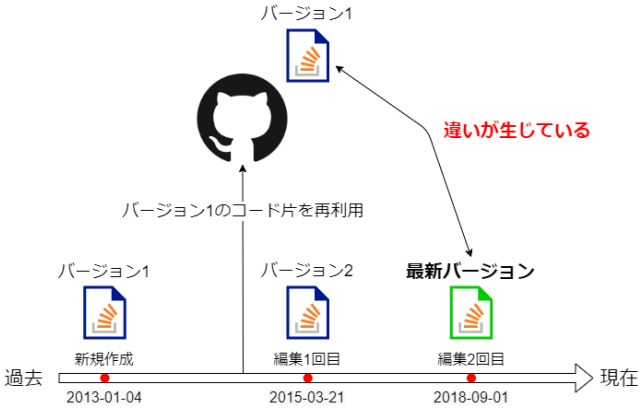

図 3 Stack Overflow の旧バージョンのコード片を利用している例

する他のソフトウェアや Q&A サイトなどからソースコードを再利用することは一般的に行われており，Stack Overflow の投稿のコード片も開発者に再利用されることが多い．Wu らは GitHub プロジェクトにおける Stack Overflow の投稿の再利用に関する調査を行い．49%の投稿が編集を伴って再利用されている事を明らかにしている [7]．また，再利用の事例において 71%のファイルが Stack Overflow の投稿への参照をソースファイル内に含んでいることも明らかにしている．

また，Stack Overflow の投稿のコード片の利用における問題も指摘されている．複数の研究で，Stack Overflow などの Q&A サイトのコード片を利用することにより，バグの混入 [8]，ライセンス違反 [9] やセキュリティの脆弱性 [10] の問題を引き起こす恐れがあることが示されている．また，Zhang らは Stack Overflow における情報の陳腐化について調査を行っている [11]．Stack Overflow において陳腐化した回答の半数以上が最初に投稿された時点で古い情報であり，そのような回答のうちその後変更されたものは 20.5%のみであった．また，陳腐化に対する指摘まで平均して 118 日要しており，陳腐化した投稿の修正には時間を要することが分かっている．

Stack Overflow のコード片を再利用する際，再利用時点から変更されるとバージョンの違いが生じることがある．具体的には Stack Overflow のコード片を GitHub プロジェクトで利用した後，参照元の Stack Overflow のコード片が変更される場合がある．そのような事例の概要を図 3 に示す．このような事例に対する調査として，Manes らの研究では Stack Overflow の投稿と，その投稿から GitHub プロジェクトで再利用されるコード片についてそれぞれの進化に着目して調査を行っている．結果として，Stack Overflow では 76%のスニペットが進化しているのに対し，GitHub では 22%しか再利用されたコード片は進化していないことがわかった．また 2 つのプラットフォームでともに進化しているものは高々 10%程度であることを明らかにしている [3]．

以上の既存研究から，Stack Overflow のコード片は GitHub プロジェクトにおいて頻繁に再利用されているが，バグや脆弱性，さらには陳腐化した情報も多い上，これらの修正には時間を要することが分かっている．これらの問題があるコード片を GitHub プロジェクトが再利用しており，Stack Overflow の投稿においてこれらの問題が修正されていた場合は，同様の修正を GitHub プロジェクトにおいても反映すべき可能性があるが，現状そのような事例は少ないことが分かる．

3. 調査

本調査では Java プロジェクトを対象として，GitHub プロジェクトに利用されている Stack Overflow のコード片の進化に関する調査を行い，進化のパターンについて分析を行う．また，これらのコード片の内容や GitHub プロジェクトにおける追従状況に関しても調査し，Stack Overflow のコード片を再利用した際に，その後のコード片の進化に追従する必要性について考察する．本研究では以下の 2 つの RQ を設定した．

- RQ1: Stack Overflow の最新バージョンのコード片を利用している GitHub プロジェクトはどの程度あるか？
- RQ2: GitHub プロジェクトに旧バージョンが利用されている Stack Overflow のコード片はどのような進化パターンがあるか？

RQ1 は GitHub プロジェクトに利用されている Stack Overflow のコード片の変更履歴を分析し，GitHub のコード片が Stack Overflow のコード片に追従しているか調査する．RQ2 は GitHub のコード片が追従していない Stack Overflow のコード片に対して進化パターンの分類を行い，進化に追従する必要性について考察する．

調査手順は以下の STEP1〜STEP4 で構成されている．

- STEP1: 既存スクリプトによるデータベース構築
- STEP2: 一回以上変更されているコード片と再利用先プロジェクトの組を抽出
- STEP3: 再利用されているコード片のバージョンの調査
- STEP4: 進化パターンの分類

STEP1 では，Baltes ら [6] が作成した SOTorrent（2018-09-23）のデータセットと Diamantopoulos ら [12] の研究で使用された既存スクリプトを使用してデータベースを構築する．STEP2 では，Diamantopoulos らが研究で使用したスクリプトを拡張し，STEP1 で構築したデータベースからコードブロックが一回以上変更されている Stack Overflow の回答とそれを利用している GitHub のファイルの URL を抽出した．STEP3 では，STEP2 で抽出した GitHub プロジェクトにおいて，Stack Overflow の最新バージョンの

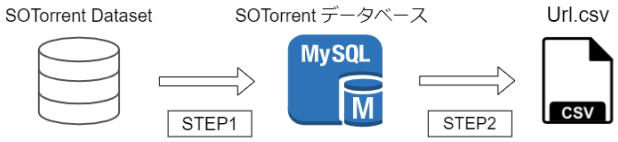

図 4　STEP1，STEP2 の概要図

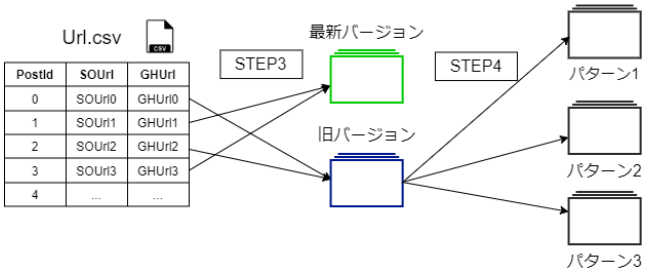

図 5　STEP3，STEP4 の概要図

コード片がどの程度利用されているかを調査した．STEP4 では，STEP3 で最新バージョンのコード片を利用していない事例において，手作業で Stack Overflow のコード片の進化パターンを利用事例における振舞いの変化に着目して進化パターンの分類を行った．

以下それぞれの STEP について詳細を述べる．

3.1　STEP1: 既存スクリプトによるデータベース構築

STEP1，STEP2 の概要図を図 4 に示す．STEP1 では SOTorrent から Java に関連した投稿のみを抽出し，データベースを構築する．本調査ではデータセットとして Baltes らの研究で作成された SOTorrent(2018-09-23) を使用し，データベース構築には Diamantopoulos らの研究で使用されたスクリプトを使用する．このスクリプトは SOTorrent の Java の投稿 (Posts テーブルの Tags カラムに “java” タグが含まれるもの) のみを抽出し，新たなデータベースを構築する．RDBMS には MySQL5.7 を用いた．

3.2　STEP2: 一回以上変更されているコード片と再利用先プロジェクトの組を抽出

STEP2 では STEP1 で構築したデータベースから，コードブロックが一回以上変更されている回答と，そのコード片を再利用している GitHub プロジェクトのファイルの URL の組を抽出した “Url.csv” を生成する．

ここで用いるスクリプトは Diamantopoulos らが研究で使用したスクリプトを拡張したものである．このスクリプトはまず，STEP1 で構築したデータベースからコードブロックが一回以上変更されている回答を抽出する．次に，その回答の URL とその回答のコード片を利用している GitHub プロジェクトのファイルの URL を抽出する．STEP1 では Tags カラムに “java”タグが含まれているもののみを抽出しているが，その場合，再利用先の GitHub プロジェクトのファイルとして Java 以外の JVM 系言語である Scala や Kotlin のファイルが含まれてしまう．そのため，抽出した GitHub プロジェクトのファイルから拡張子が “java” 以外のものは除外するようにスクリプトを拡張した．最後に Stack Overflow の回答の ID（PostId），Stack Overflow の回答の URL（SOUrl），回答のコード片を再利用している GitHub プロジェクトのファイルの URL（GHUrl）の 3 つのカラムを含む “Url.csv” を生成する．

3.3　STEP3: 再利用されているコード片のバージョンの調査

STEP3，STEP4 の概要図を図 5 に示す．STEP2 で生成された “Url.csv” から GitHub プロジェクトで Stack Overflow のコード片の最新バージョンを利用しているものはどの程度あるか調査を行う．本調査における Stack Overflow のコード片のバージョンの変更とは，Stack Overflow のコード片が再利用された GitHub プロジェクトのコミット以降，再利用された コード片と関連するコードブロックの変更があった場合を指す．本調査では，同じプロジェクトからフォークあるいは複製されたプロジェクトの影響を除外するため，Stack Overflow から複数の GitHub プロジェクトに利用されている場合，Stack Overflow の投稿 1 件につき，1 つの GitHub プロジェクトに絞り調査を行った．GitHub プロジェクトが Stack Overflow のコード片の最新バージョンを利用しているものを「最新バージョン」，それ以外のバージョンのコード片を利用しているものを「旧バージョン」と定義する．なお，本調査における最新バージョンとは，2021 年 5 月 18 日時点における最新のバージョンとする．以上の定義に基づき，STEP2 で抽出した “Url.csv” をもとに，GitHub プロジェクトが Stack Overflow のコード片の最新バージョンを利用しているか旧バージョンを利用しているかで 2 つに分類した．

3.4　STEP4: 進化パターンの分類

次に STEP3 で旧バージョンを利用していると分類されたものに対して，GitHub プロジェクトが Stack Oveflow のコード片を再利用した後に Stack Overflow のコード片が変更された場合，どのような進化パターンがあるか調査を行う．進化パターンは，Stack Overflow のコード片の変更により GitHub プロジェクトに与える影響の有無に基づいて分類を行った．以下に分類した 3 つの進化パターンに関して示す．

- パターン 1: GitHub プロジェクトで再利用されたコード片が Stack Overflow のコード片の進化に追従する際，振舞いが変化する変更
- パターン 2: GitHub プロジェクトで再利用されたコード片が Stack Overflow のコード片の進化に追従する際，振舞いが変化しない変更

- パターン 3: Stack Overflow の説明を補足するものであり，GitHub プロジェクトに直接影響しない変更

本調査における “振舞いが変化する” とは，GitHub プロジェクトで再利用されたコード片が Stack Overflow のコード片の進化に追従した場合に，開発者から見てプロジェクトに影響が生じるか，である．

パターン 1 に分類される振舞いが変わる例としてはバグ修正が挙げられる． 進化に追従することでバグが修正され，エラーが発生して正常に実行できなかったプロジェクトやメソッドが実行可能になる場合，開発者からみてプロジェクトに影響があり，振舞いが変化していることになる．パターン 2 に分類される振舞いが変化しない例としてはリファクタリングが挙げられる．リファクタリングはソースコードの内部構造を変更するだけでプロジェクトやメソッドの振舞いは変化しない．パターン 3 は Stack Overflow の説明を補足するものであり，プロジェクトやメソッドに直接影響を与えない変更である．例えば，import 文を書き加えるような変更は，Stack Overflow 上のコード片で不足していたものを補う変更であるが，そのコード片を GitHub などで再利用する際には通常記述済みのはずである．

4. 調査結果

4.1 RQ1: Stack Overflow の最新バージョンのコード片を利用している GitHub プロジェクトはどの程度あるか？

3 章で説明した STEP1 から STEP3 の手順に従って得られた，コードブロックが一回以上変更されているコード片とそのコード片を再利用している GitHub プロジェクトのファイルの組は 263 件であった．これらの組に対して，再利用されているコード片が最新バージョンであるか旧バージョンであるかを分類した．その結果を表 1 に示す．

263 件中 31 件についてはプライベートリポジトリに変更された，または，GitHub のリポジトリが削除されたため，Stack Overflow のコード片のどのバージョンを利用しているか調査できなかった．調査可能な 232 件のうち，186 件が最新バージョンを利用しており 46 件が旧バージョンを利用していた．つまり全体の 20%の事例で，Stack Overflow の旧バージョンのコード片が利用されていることが分かる．

4.2 RQ2: GitHub プロジェクトに旧バージョンが利用されている Stack Overflow のコード片はどのような進化パターンがあるか？

GitHub プロジェクトが再利用した Stack Overflow のコード片が再利用後に変更された場合，コード片の進化パターンを開発者から見たプロジェクトの振舞いの変化を基に 3.4 節で示した 3 つのパターンに分類した．調査対象は STEP3 で Stack Overflow の旧バージョンのコード片を利用していた 46 件である．表 2 に進化パターンの分類結果を示す．

表 1 再利用されているコード片のバージョンの調査結果

利用状況	件数 (263 件)
最新バージョン	186
旧バージョン	46
リンクが無効	31

表 2 進化パターンの分類結果

進化パターン	件数 (46 件)
パターン 1（振舞い変更あり）	17
パターン 2（振舞い変更なし）	25
パターン 3（投稿の説明補足）	4

パターン 1 は Stack Overflow の進化に追従した場合に開発者から見て GitHub プロジェクトに影響を与えるものであり，17 件であった．事例としてはバグ修正，例外処理，境界値の変更などがある．それに対してパターン 2 は GitHub プロジェクトに影響を与えないものであり 25 件であった，ソフトウェアの品質向上や処理速度の高速化などの意図で変更されており，例としてはリファクタリング，機能向上，型変更などがある．例外としてメソッド名の誤字修正は変更することで GitHub プロジェクトの振舞いが変化してしまうが，ソフトウェアの品質に関する変更であり，メソッド自体の振舞いが変わっているわけではないのでパターン 2 に分類した．パターン 3 は 4 件あり，例としては，import 文やクラス宣言を追加する説明補足などあった．

それぞれの進化パターンについて例を示し，どのような進化パターンであれば進化に追従する必要性があるか考察した．コード片の行頭に「+」がついているものはその変更により追加された行であり，「-」がついているものはその変更により削除された行である．

4.2.1 パターン 1

バグ修正 投稿 ID：7322581

Java を用いた大きなファイルからテキストの最終行を高速で読み取る方法についての質問に対して，回答として以下のようなコードブロックが寄せられた．

```
public String tail2( File file, int lines) {
        ...
            if( readByte == 0xA ) {
-                line = line + 1;
-                if (line == lines) {
-                    if (filePointer == fileLength) {
-                        continue;
-                    }
-                    break;
+                if (filePointer < fileLength) {
+                    line = line + 1;
+                }
            }
            } else if( readByte == 0xD ) {
-                line = line + 1;
-                if (line == lines) {
```

```
-                   if (filePointer == fileLength - 1) {
-                       continue;
-                   }
-                   break;
-               }
+               if (filePointer < fileLength-1) {
+                   line = line + 1;
+               }
+           }
+           if (line >= lines) {
+               break;
            }
            sb.append( ( char ) readByte );
        ...
}
```

変更前のコード片では改行コード \r \n のあるファイルでは正しく機能しない．改行コード \r \n を使用すると，行数をカウントする条件 line == lines をスキップして全体を読み取ることになる．よって改行コード \r \n の場合，1 行ごとに 2 行カウントされてしまうので，変更後のコード片では，最終行の N 行を取得するために，n*2 行を指定するように変更されている．これはコード片を利用する際に発生するバグの修正を行なっており変更によって振舞いが変化しているため，パターン 1 に分類される．

例外処理 投稿 ID：35989142

Java.nio パッケージを使用して最も簡単に再帰的にディレクトリを削除する方法についての質問に対して，回答として以下のようなコードブロックが寄せられた．

```
Path rootPath = Paths.get("/data/to-delete");
// before you copy and paste the snippet
        ...
// the snippet below
+ try (Stream<Path> walk = Files.walk(rootPath)) {
+     walk.sorted(Comparator.reverseOrder())
    Files.walk(rootPath)
        .sorted(Comparator.reverseOrder())
        .map(Path::toFile)
        .peek(System.out::println)
        .forEach(File::delete);
+ }
```

ここではファイルシステムのリソースをタイムリーに廃棄する必要があるため，例外処理として try-with-resources 文が追加されている．このコード片の進化に追従することで GitHub プロジェクトの振舞いが変化するため，パターン 1 に分類される．

4.2.2 パターン 2

リファクタリング 投稿 ID：3775723

Java を使用してファイルを削除する方法の質問に対して，回答として以下のようなコードブロックが寄せられた．

```
public static boolean deleteDir(File dir) {
 if (dir.isDirectory())
 {
  String[] children = dir.list();
  for (int i=0; i<children.length; i++)
- {
-  boolean success = deleteDir(new File(dir, children[i]));
-  if (!success)
-  {
-   return false;
-  }
- }
+ return deleteDir(new File(dir, children[i]));
 }
// The directory is now empty or this is a file so delete it
 return dir.delete();
}
```

変更前は boolean 型のメソッドの返り値を変数に代入して条件分岐に使っていたが，変更後は変数への代入を行わずに処理を行っている．これはコード片のリファクタリングを行っており，変更によって振舞いが変化することはない．よってパターン 2 に分類される．

機能向上 投稿 ID：2222268

サーブレットに対して送られてきたリクエストに関して，正確な URL を取得するには HttpServletRequest インターフェースのオブジェクトを使用する．その使用方法の質問に対して，回答として以下のようなコードブロックが寄せられた．

```
public static String getFullURL(HttpServletRequest request){
-   StringBuffer requestURL = request.getRequestURL();
+   StringBuilder requestURL = new StringBuilder(request
+   .getRequestURL().toString());
    String queryString = request.getQueryString();

    if (queryString == null) {
        return requestURL.toString();
    } else {
        return requestURL.append('?').append(queryString)
        .toString();
    }
}
```

ここでは，StringBuffer を StringBuilder に変更している．StringBuffer はスレッドセーフであり，StringBuilder はスレッドセーフではない．コード片を利用しているプロジェクトでは，この変更によりパフォーマンスが向上するが，振舞いは変化しないためパターン 2 に変更される．

パターン 2 についてリファクタリングやパフォーマンスの向上など，ソースコードの振舞いは変化しないが，ソースコードの可読性の観点から進化に追従する必要性はあると考察する．

4.2.3 パターン 3

説明補足 投稿 ID：10174938

指定された文字列が Java で有効な JSON であるかどうかを確認する方法の質問に対して，回答として以下のようなコードブロックが寄せられた．

```
+ import org.json.*;
public boolean isJSONValid(String test) {
    try {
        new JSONObject(test);
    } catch (JSONException ex) {
        // edited, to include @Arthur's comment
        // e.g. in case JSONArray is valid as well...
        try {
            new JSONArray(test);
        } catch (JSONException ex1) {
            return false;
        }
    }
    return true;
}
```

変更点としては，import 文が追加されている．ここでは GitHub，maven，Android で利用可能な org.json JSONAPI 実装を使用するように説明補足されている．これは開発者がよりこのコード片を利用しやすいように変更されているが，コード片を利用しているプロジェクトには直接影響しないものであり，パターン 3 に分類される．

クラス宣言の追加 投稿 ID：2248203

Java の ThreadPoolExecutor クラスを使用して，高負荷のタスクを固定数のスレッドで実行する際の例外処理の方法に関する質問に対して，回答として以下のようなコードブロックが寄せられた．

```
+ public final class ExtendedExecutor
+ extends ThreadPoolExecutor {
+     // ...
        ...
    protected void afterExecute(Runnable r, Throwable t) {
          super.afterExecute(r, t);
        ...
    }
+ }
```

ここでは，メソッドがどのクラスを継承することで利用可能かを開発者に分かりやすく示すために，クラス宣言が追加されている．このコード片を参照している GitHub プロジェクトにおいても該当クラスの継承を利用しており，GitHub プロジェクトに直接影響する変更ではないためパターン 3 に分類される．

4.3 考察

RQ2 では GitHub プロジェクトで Stack Overflow の旧バージョンが使われている 46 件の中で，パターン 1 に分類される 17 件は Stack Overflow の投稿の進化に追従することで GitHub プロジェクトの振舞いが変化することが分かった．パターン 1 は振舞いが変化するためすべての進化に追従することはできないが，バグ修正や例外処理など Stack Overflow のコード片の処理を正常化するための変更がある．そのため，開発者は GitHub プロジェクトが利用している Stack Overflow のコード片の進化を追い，その変更内容を把握し，必要に応じて追従すべきである．パターン 2 は GitHub プロジェクトが Stack Overflow のコード片の進化に追従しても振舞いは変化しないため，パターン 1 に比べて即座に進化に追従する必要性は薄い．しかし，リファクタリングやパフォーマンスの向上などのソフトウェアの品質面における修正が多いため，進化に追従する価値はあると考えられる．パターン 3 は説明補足のための変更であるため開発者は変更履歴を追う必要はない．

5. まとめと今後の課題

本研究では，GitHub プロジェクトが Stack Overflow のコード片の進化に追従する必要性を調べるため，実際の再利用の事例に基づいて詳細な分析を行った．具体的にはデータセットとして SOTorrent を用いて GitHub プロジェクトに利用されている Stack Overflow のコード片の変更履歴を分析し進化パターンの分類を行い，進化に追従する必要性について考察した．その結果，調査可能な 232 件のうち 186 件が Stack Overflow の最新バージョンのコード片を利用しており，46 件が旧バージョンのコード片を使用していた．旧バージョンのうち GitHub プロジェクトが Stack Overflow のコード片の進化に追従することで振舞いが変化するもの（パターン 1）が 17 件，振舞いが変化しないもの（パターン 2）が 25 件，振舞いに直接影響しないもの（パターン 3）が 4 件であった．この中でも特にパターン 1 では，バグ修正などでコード片が進化しているものがあるため，開発者は GitHub プロジェクトが利用している Stack Overflow のコード片の進化を追い，その変更内容を把握し，必要に応じて追従すべきである．

今後の課題として別のプログラミング言語への調査対象の拡張があげられる．本調査では対象を GitHub で公開されている Java プロジェクトに限定したが，プログラミング言語は人気や歴史が異なるため，最新バージョンと旧バージョンの件数など結果に偏りが生じる可能性がある．特に特定のタグ（node.js，ajax，Objective-c）に関連する質問への回答は陳腐化する傾向がある [11] ため，これらのタグに関しても調査を行う必要がある．

謝辞 本研究は JSPS 科研費 JP18H04094, JP19K20239 の助成を受けたものです．

参考文献

[1] Hata, H., Treude, C., Kula, R. G. and Ishio, T.: 9.6 Million Links in Source Code Comments: Purpose, Evolution, and Decay, *Proceedings of the 41st International Conference on Software Engineering (ICSE)*, pp. 1211–1221 (2019).
[2] Manes, S. S. and Baysal, O.: How Often and What StackOverflow Posts Do Developers Reference in Their

GitHub Projects?, *Proceedings of the 16th International Conference on Mining Software Repositories (MSR)*, pp. 235–239 (2019).

[3] Manes, S. S. and Baysal, O.: Studying the Change Histories of Stack Overflow and GitHub Snippets, *Proceedings of the 18th International Conference on Mining Software Repositories (MSR)*, pp. 283–294 (2021).

[4] StackExchangeInc: Stack Overflow, `https://stackoverflow.com/`.

[5] StackExchangeInc: All Sites - Stack Exchange, `https://stackexchange.com/sites?view=list#users`(accessed on 2021-06-07).

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

[7] Wu, Y., Wang, S., Bezemer, C.-P. and Inoue, K.: How Do Developers Utilize Source Code from Stack Overflow?, *Empirical Software Engineering*, Vol. 24 (online), DOI: 10.1007/s10664-018-9634-5 (2019).

[8] Ragkhitwetsagul, C., Krinke, J., Paixao, M., Bianco, G. and Oliveto, R.: Toxic Code Snippets on Stack Overflow, *IEEE Transactions on Software Engineering*, Vol. 47, No. 3, pp. 560–581 (2021).

[9] An, L., Mlouki, O., Khomh, F. and Antoniol, G.: Stack Overflow: A code laundering platform?, *Proceedings of the 24th International Conference on Software Analysis, Evolution and Reengineering (SANER)*, pp. 283–293 (2017).

[10] Fischer, F., Böttinger, K., Xiao, H., Stransky, C., Acar, Y., Backes, M. and Fahl, S.: Stack Overflow Considered Harmful? The Impact of Copy Paste on Android Application Security, *Proceedings of the Symposium on Security and Privacy (SP)*, pp. 121–136 (2017).

[11] Zhang, H., Wang, S., Chen, T.-H., Zou, Y. and Hassan, A. E.: An Empirical Study of Obsolete Answers on Stack Overflow, *IEEE Transactions on Software Engineering*, Vol. 47, No. 4, pp. 850–862 (2021).

[12] Diamantopoulos, T., Sifaki, M. I. and Symeonidis, A.: Towards Mining Answer Edits to Extract Evolution Patterns in Stack Overflow, *Proceedings of the 16th International Conference on Mining Software Repositories (MSR)*, pp. 215–219 (2019).