

Analyzing and Supporting Adaptation of Online Code Examples

Tianyi Zhang, Di Yang, Crista Lopes, Miryung Kim

In Proceedings of the 41st International
Conference on Software Engineering (ICSE '19)
316-327

背景

- Stack Overflowを代表とするQAサイトの回答ではコードの例が載せられる
- コード例を自分のコードに適用したい開発者はコード例を使える形に直す必要がある
- コード例を直すには手間がかかる

目的・アプローチ

- 目的

- コード例を自分のコードに適用する際の援助を行う
ツールEXAMPLESTACKを構築する

- アプローチ

- SOのコード例の内、GitHubのプロジェクトで利用されているものを見つける
- SOのコード例がGitHubのプロジェクトでどのように適用されているかを調査する
- 調査した適用タイプからツールを構築する

SOからコード例の抽出

- 2016年10月に投稿された回答を使用
- タグがjavaまたはandroidの回答312219本の中のhtmlタグ<code>で囲まれたコードを取得



利用されたSOのコード例の特定

1. GitHubプロジェクトの選択

- フォークされたものではない, かつ, スターが5つ以上

2. GitHubプロジェクトで利用されているSOコードの特定

- クローン検出ツールSourcererCC[1]を使用

3. GitHubコードとSOコードの作成日時と比較

- GitHubコードよりも後に作成されたSOコード例を除去

4. GitHubコードでSOの投稿の参照があるかを確認

- GitHubコードにSOのコード例を参考にした旨を明記しているかを確認

タイプ分けに用いるデータ

- 利用されたコード例特定step4まで
 - adaptationデータセット
 - 629件
- 利用されたコード例特定step3まで
 - variationデータセット
 - 14124件
- それぞれのデータセットから200件ずつ適用タイプの分類に用いる

SOコードとGitHubコードの差分取得

- コード適用時に加えた変更を取得するために差分取得を行う
- **GumTree[2]**を用いてコードを**AST**に変換し，差分を取得
 - **GumTree**：ソースコードを**AST**に変換し，差分抽出などを行うツール

適用タイプの分類

- 本論文の2人の筆者がコード例を目視で適用タイプを分類
- 200のコード片の内、まず90個ずつ目視して適用タイプが存在することを観察
- 残りのコード片も分類

得られた適用タイプ

TABLE I: Common adaptation types, categorization, and implementation

Category	Adaptation Type	Rule
Code Hardening	Add a conditional	$\text{Insert}(t_1, t_2, i) \wedge \text{NodeType}(t_1, \text{IfStatement})$
	Insert a final modifier	$\text{Insert}(t_1, t_2, i) \wedge \text{NodeType}(t_1, \text{Modifier}) \wedge \text{NodeValue}(t_1, \text{final})$
	Handle a new exception type	$\text{Exception}(e, \text{GH}) \wedge \neg \text{Exception}(e, \text{SO})$
	Clean up unmanaged resources (e.g. close a stream)	$(\text{LocalCall}(m, \text{GH}) \vee \text{InstanceCall}(m, \text{GH})) \wedge \neg \text{LocalCall}(m, \text{SO}) \wedge \neg \text{InstanceCall}(m, \text{SO}) \wedge \text{isCleanMethod}(m)$
Resolve Compilation Errors	Declare an undeclared variable	$\text{Insert}(t_1, t_2, i) \wedge \text{NodeType}(t_1, \text{VariableDeclaration}) \wedge \text{NodeValue}(t_1, v) \wedge \text{Use}(v, \text{SO}) \wedge \neg \text{Def}(v, \text{SO})$
	Specify a target of method invocation	$\text{InstanceCall}(m, \text{GH}) \wedge \text{LocalCall}(m, \text{SO})$
	Remove undeclared variables or local method calls	$(\text{Use}(v, \text{SO}) \wedge \neg \text{Def}(v, \text{SO}) \wedge \neg \text{Use}(v, \text{GH})) \vee (\text{LocalCall}(m, \text{SO}) \wedge \neg \text{LocalCall}(m, \text{GH}) \wedge \neg \text{InstanceCall}(m, \text{GH}))$
Exception Handling	Insert/delete a try-catch block	$(\text{Insert}(t_1, t_2, i) \vee \text{Delete}(t_1)) \wedge \text{NodeType}(t_1, \text{TryStatement})$
	Insert/delete a thrown exception in a method header	$\text{Changed}(t_1) \wedge \text{NodeType}(t_1, \text{Type}) \wedge \text{Parent}(t_2, t_1) \wedge \text{NodeType}(t_2, \text{MethodDeclaration}) \wedge \text{NodeValue}(t_1, t) \wedge \text{isExceptionType}(t)$
	Update the exception type	$\text{Update}(t_1, t_2) \wedge \text{NodeType}(t_1, \text{SimpleType}) \wedge \text{NodeType}(t_2, \text{SimpleType}) \wedge \text{NodeValue}(t_1, v_1) \wedge \text{isExceptionType}(v_1) \wedge \text{NodeValue}(t_2, v_2) \wedge \text{isExceptionType}(v_2)$
	Change statements in a catch block	$\text{Changed}(t_1) \wedge \text{Ancestor}(t_2, t_1) \wedge \text{NodeType}(t_2, \text{CatchClause})$
	Change statements in a finally block	$\text{Changed}(t_1) \wedge \text{Ancestor}(t_2, t_1) \wedge \text{NodeType}(t_2, \text{FinallyBlock})$
Logic Customization	Change a method call	$\text{Changed}(t_1) \wedge \text{Ancestor}(t_2, t_1) \wedge \text{NodeType}(t_2, \text{MethodInvocation})$
	Update a constant value	$\text{Update}(t_1, t_2) \wedge \text{NodeType}(t_1, \text{Literal}) \wedge \text{NodeType}(t_2, \text{Literal})$
	Change a conditional expression	$\text{Changed}(t_1) \wedge \text{Ancestor}(t_2, t_1) \wedge (\text{NodeType}(t_2, \text{IfCondition}) \vee \text{NodeType}(t_2, \text{LoopCondition}) \vee \text{NodeType}(t_2, \text{SwitchCase}))$
	Change the type of a variable	$\text{Update}(t_1, t_2) \wedge \text{NodeType}(t_1, \text{Type}) \wedge \text{NodeType}(t_2, \text{Type})$
Refactoring	Rename a variable/field/method	$\text{Update}(t_1, t_2) \wedge \text{NodeType}(t_1, \text{Name})$
	Replace hardcoded constant values with variables	$\text{Delete}(t_1) \wedge \text{NodeType}(t_1, \text{Literal}) \wedge \text{Insert}(t_1, t_2, i) \wedge \text{NodeType}(t_1, \text{Name}) \wedge \text{Match}(t_1, t_2)$
	Inline a field	$\text{Delete}(t_1) \wedge \text{NodeType}(t_1, \text{Name}) \wedge \text{Insert}(t_1, t_2, i) \wedge \text{NodeType}(t_1, \text{Literal}) \wedge \text{Match}(t_1, t_2)$
Miscellaneous	Change access modifiers	$\text{Changed}(t_1) \wedge \text{NodeType}(t_1, \text{Modifier}) \wedge \text{NodeValue}(t_1, v) \wedge v \in \{\text{private}, \text{public}, \text{protected}, \text{static}\}$
	Change a log/print statement	$\text{Changed}(t_1) \wedge \text{NodeType}(t_1, \text{MethodInvocation}) \wedge \text{NodeValue}(t_1, m) \wedge \text{isLogMethod}(m)$
	Style reformatting (i.e., inserting/deleting curly braces)	$\text{Changed}(t_1) \wedge \text{NodeType}(t_1, \text{Block}) \wedge \text{Parent}(t_2, t_1) \wedge \neg \text{Changed}(t_2) \wedge \text{Child}(t_3, t_1) \wedge \neg \text{Changed}(t_3)$
	Change Java annotations	$\text{Changed}(t_1) \wedge \text{NodeType}(t_1, \text{Annotation})$
	Change code comments	$\text{Changed}(t_1) \wedge \text{NodeType}(t_1, \text{Comment})$

適用に要する変更数の調査

- **SO**のコード例を自分のコードに適用する際に加える変更数を調査

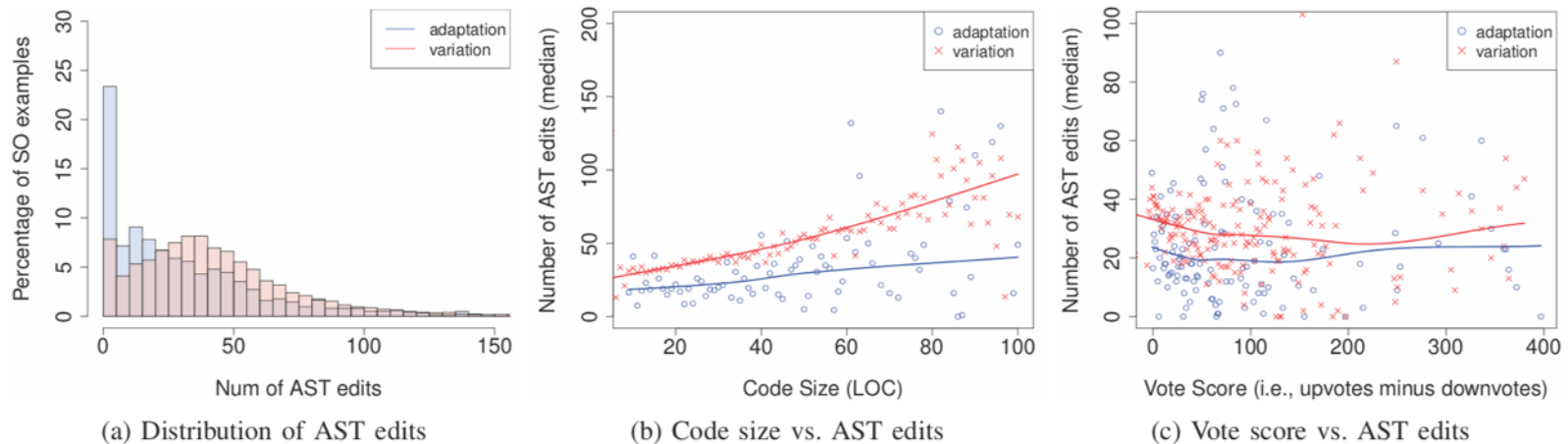
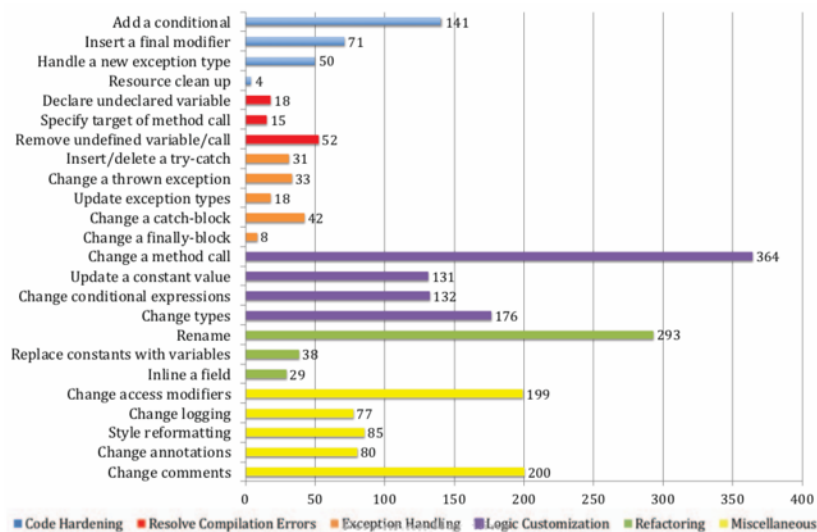


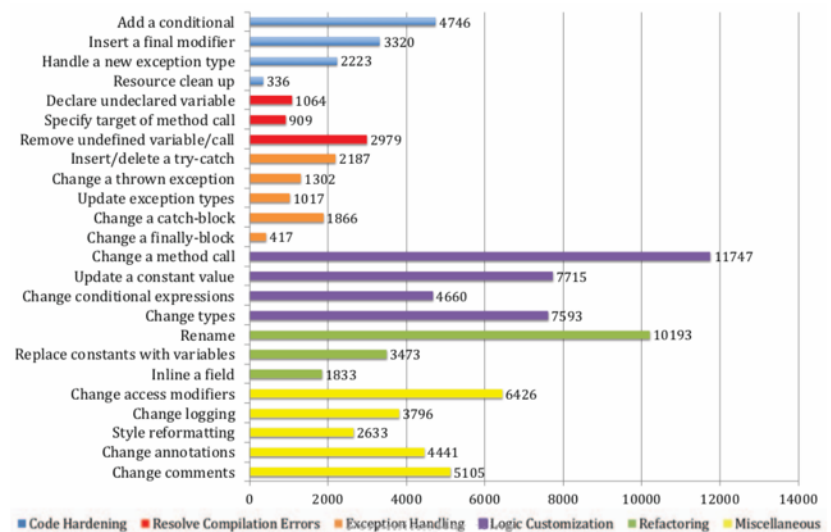
Fig. 2: Code size (LOC) and vote scores on the number of AST edits in a SO example

適用タイプの頻度の調査

- adaptationデータセットとvariationデータセットとで用いられた適用タイプの頻度を調査



(a) Adaptations: 629 explicitly attributed SO examples



(b) Variations: 14,124 potentially reused SO examples

Fig. 3: Frequencies of categorized adaptation types in two datasets

EXAMPLESTACKの動作

1. SOから利用したいコードを見つける
2. GitHubから類似のコードを取得
3. 虫食いコードの生成
4. 虫食い部分を自分のコードに合うように埋める

EXAMPLESTACKの動作

- Chromeの拡張機能として動作する

The screenshot displays the ExampleStack Chrome extension interface. On the left, a Stack Overflow question is shown with a code snippet for reading JSON data from an assets file. A red box highlights a specific code example, labeled ①. On the right, the 'Code Template' panel shows the same code snippet with common unchanged code retained and adapted regions abstracted with *hot spots*. A red box highlights the template, labeled ②. Below the template, a list of similar GitHub snippets is shown, with a red box highlighting a specific snippet, labeled ③. The snippet includes a GitHub link and metrics, labeled ④. The interface also includes a 'Copy the template' button, labeled ⑤, and a 'Copy' button, labeled ⑥. The adaptation categories are listed on the right, labeled ⑦.

stackoverflow Questions Developer Jobs Tags Users Search...

Adding on the @Pso's comment, you can store all your Lat,Lng values in a JSON file and copy that file to the assets folder of your app.

1 So, let's say you save the data like this in `assets/locations.json` file,

```
{
  "data": [
    [-08.8123003, 13.2249500],
    [-08.8265861, 13.2274667],
    [-08.8328611, 13.2182861],
    ....]
}
```

And then read it as, ① A Stack Overflow code example of interest

```
public String getJSONFromAssets() {
    String json = null;
    try {
        InputStream inputData = getAssets().open("locations.json");
        int size = inputData.available();
        byte[] buffer = new byte[size];
        inputData.read(buffer);
        inputData.close();
        json = new String(buffer, "UTF-8");
    } catch (IOException ex) {
        ex.printStackTrace();
        return null;
    }
    return json;
}
```

Welcome to ExampleStack! ⑥ Copy the template

Code Template ⑤ Undo the previous selection Undo Selection Copy

```
String loadJSONFromAssets(String jsonFileName) {
    String json = null;
    try {
        InputStream is = getAssets().open(jsonFileName);
        int size = is.available();
        byte[] buffer = new byte[size];
        is.read(buffer);
        is.close();
        json = new String(buffer, "UTF-8");
    } catch (IOException ex) {
        ex.printStackTrace();
        return null;
    }
    return json;
}
```

Adaptation Categories:

- Code Hardening
- Resolve Compilation Error
- Exception Handling
- Logic Customization
- Refactoring
- Miscellaneous

⑦ Colors of different adaptation categories

5 Similar GitHub Examples ③ A list of similar GitHub snippets

GitHub link watch: 27 star: 197 contributor: 15 ④ GitHub snippet link & metrics

```
public String loadJSONFromAssets(String jsonFileName) {
    String json = null;
    try {
        InputStream is = getAssets().open(jsonFileName);
        int size = is.available();
        byte[] buffer = new byte[size];
        is.read(buffer);
        is.close();
        json = new String(buffer, "UTF-8");
    } catch (IOException ex) {
        ex.printStackTrace();
        return null;
    }
    return json;
}
```

Fig. 4: In the lifted template, common unchanged code is retained, while adapted regions are abstracted with *hot spots*.

EXAMPLESTACKの動作

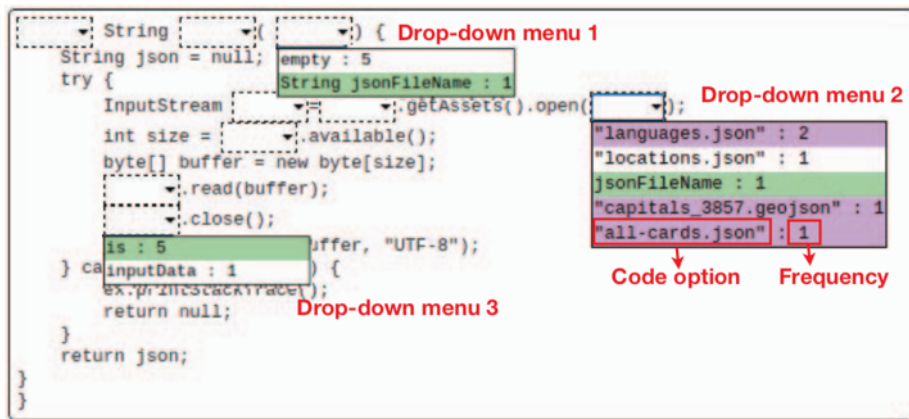


Fig. 5: Alice can click on a hot spot and view potential code options colored based on their underlying adaptation type.

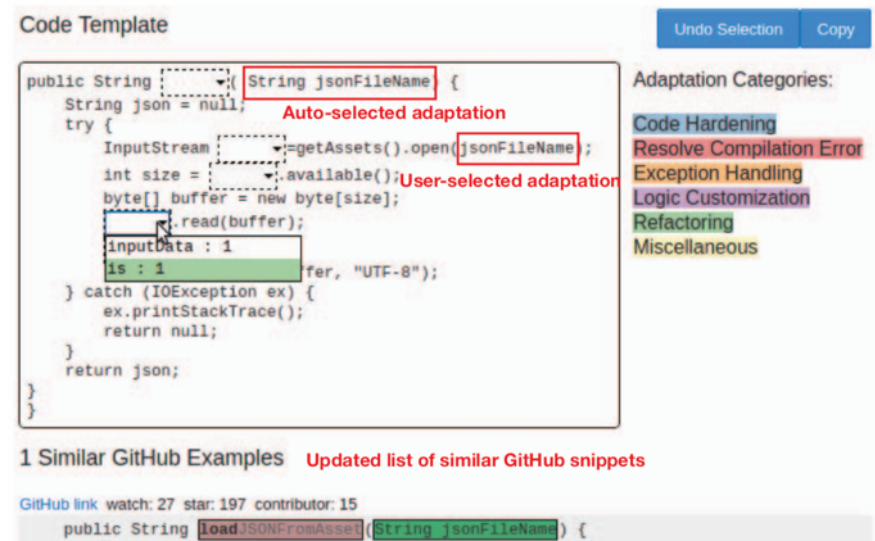


Fig. 6: EXAMPLESTACK automatically updates the code template based on user selection.

虫食いコードの生成

- SOコードとGitHubコードの差分取得
 - 差分がある部分を空欄にする
- 差分のグルーピング
 - 同じ位置にあり、意味的に異ならない部分を1つの差分にまとめる
- ドロップダウンメニューの作成と色つけ
 - 複数の差分をドロップダウンに表示
 - 各差分がどの適用カテゴリに属するかを判定し色付けする

ツールの評価実験

- ソフトウェア工学を学ぶ学部生14人と大学院生2人がツールの評価
- ツールを使う場合と使わない場合とで4種のタスクを用意し、操作性を比較
- 各タスクでどの部分を変更したか、またどのようにして変更したかを尋ねる
- 提案したツールが開発者に適用方法を思いつかせることができるかを評価する

評価実験の結果

全体的に加えられた変更数が多く手間がかかっているが、コードの安全性の考慮やロジックの変更の追加が行われている

TABLE II: Code reuse tasks and user study results

ID	Desired Function & SO Example	LOC	Clone#	Control			Experiment		
				Assignment	Adaptation	Time(s)	Assignment	Adaptation	Time(s)
Task I	Calculate the geographic distance between two GPS coordinates [45]	12	2	P5-A	refactor(5), logic(1)	458	P2-A	harden(1), logic(1), misc(2)	870
				P7-A	refactor(1), logic(2), misc(1)	900	P3-B	refactor(6), logic(4), misc(3)	900
				P12-B	refactor(2), harden(1)	900	P10-B	refactor(5), logic(2), misc(1)	366
				P16-B	refactor(7)	727	P15-A	refactor(10), logic(14), misc(3)	842
Task II	get the relative path between two files [46]	74	2	P3-A	refactor(5), logic(1), exception(2), misc(3)	900	P1-B	refactor(3), harden(1), logic(2)	640
				P8-A	harden(1)	900	P6-A	harden(4), logic(3)	900
				P11-B	none	621	P9-A	harden(4), logic(2)	900
				P15-B	refactor(13), harden(1), logic(5), exception(1), misc(1)	863	P13-B	refactor(3), logic(2), exception(1), misc(1)	900
Task III	encode a byte array to a hexadecimal string [47]	12	17	P1-A	refactor(5), harden(1)	652	P4-A	refactor(5), harden(1), misc(1)	667
				P6-B	refactor(1), misc(1)	900	P8-B	refactor(2), harden(1), misc(2)	548
				P9-B	harden(1), logic(1)	635	P12-A	refactor(3), harden(2), misc(1)	748
				P13-A	refactor(3), misc(1)	900	P14-B	refactor(3), harden(1), misc(1)	700
Task IV	add animation to an Android view [48]	29	4	P2-B	refactor(3), logic(1)	441	P5-B	refactor(1), logic(3)	478
				P4-B	refactor(1), compile(1), misc(1)	900	P7-B	refactor(2), compile(3), logic(3)	887
				P10-A	refactor(3), logic(5)	900	P11-A	refactor(1), logic(3)	617
				P14-A	refactor(2), logic(4)	862	P16-A	refactor(6), logic(4), misc(1)	773

ツールを使わない

ツールを使う

参加者の声

- SOコードを適用する際にGitHubコードを見た方が適用しやすいか
 - 適用する際の納得を得るための近道になる
 - ミスするかもしれない部分に注目できる
- 虫食いコードでのやり取りは適切か
 - 従うべき良い習慣を強調し，最初にすべき変更を優先づけしてくれる

参加者の声

- カテゴリごとの色分けは適切か
 - 多くの被験者は便利だと答えたが6人の被験者は混乱すると訴えた
- どのようなときにこのツールを使うか
 - APIの使い方を学ぶとき
 - 規模が大きいコードを適用したいとき

まとめ・所感

- まとめ

- コード例を自分のコードに適用する際のタイプを分類した
- どのような適用タイプが頻出するかを調査した
- 適用タイプをもとにコード例の適用の援助を行うツールを実装した

- 所感

- ツールを使った場合のコードの適用の方が後々ソースコードの品質に悪影響を与えにくいかが気になる