How Different Is It Between Machine-Generated and Developer-Provided Patches?

An Empirical Study on The Correct Patches Generated by Automated Program Repair Techniques

Shangwen Wang，Ming Wen，Liqian Chen，Xin Yi，Xiaoguang Mao

マシンが作成したパッチと開発者が作成したパッチの違いについて

プログラム自動修正技術によって生成された正しいパッチに関する実証研究

発表者：辻　裕真

2019 ACM/IEEE International Symposium on Empirical Software Engineering and Measurement (ESEM)

背景

- 自動プログラム修正(Automated Program Repair，APR)はバグの修正コストを削減できるため学界，産業界の両方で多くの注目を集めてきたが，自動で作成されたパッチの正確さを評価する方法に関しては確立されていない.
- 既存のAPR技術によって生成された正しいパッチに関する実証研究を実施し，その特徴を理解することが目的.

パッチの評価

- 生成されたパッチが正しいかどうかはテストケースに合格するかどうかで判断されているが，実際問題として全てのテストケースに合格していても正しいとは言えないパッチが数多く存在することが指摘されている.
- その対策として，開発者が手動でパッチの正確性をチェックするという方法があり，高い効果が期待できるが開発者の主観が大きく影響するという問題もある.

Research Questions

RQ1：マシンが生成した正しいパッチは，開発者が提供したものとどのような違いがあるか.

RQ2：それぞれのタイプのパッチはどのように分布しているか.

RQ3：それぞれのタイプのパッチの特徴について.

Research Questions

RQ1：マシンが生成した正しいパッチは，開発者が提供したものとどのような違いがあるか.

- それぞれのパッチの違いは修正された場所と変更内容の2つの観点から調査を行う.
- 収集したパッチをSLSM，SLDM，DLSM，DLDMの4つのタイプに分類しそれぞれの特徴について調査する.

(Same，Different，Location，Modification)

手法

- Defects4Jというバグデータセットは6つのオープンソースプロジェクト(JFreechart, Closure compiler, Apache commons-lang, Apache commons-math, Mockito, JodaTime)から抽出された395のバグとパッチのセットが含まれている.

- この中から10の一般的なARP手法(ACS, ssFix, JAID, CapGen, Elixir, SimFix, AVATAR, Nopol, jGenProg, jKail)によって生成された185のパッチを収集.(これらのパッチは開発者によって正しいと判断されている)

手法

- 185のパッチが本当に正しいものであるかを確認するために,
  27人の大学生に各パッチが正しいかどうかを判断してもらう.
- 過半数の参加者に正しくないと判断されたパッチを除外して,
  177のパッチを調査の対象とする.

結果

RQ1：マシンが生成した正しいパッチは，開発者が提供したものとどのような違いがあるか.

SLSM(132，74.6%)

- このタイプはマシンが生成したパッチと開発者が提供したパッチが同一であることを示し，最も一般的であった.

SLDM(33，18.6%)

- 開発者が提供したパッチと比較してどのように意味的に同等なパッチが作成されたかを表3に示す.

- 開発者に比べマシンが生成するパッチには条件文に関する変更が多いことが分かった.

結果

RQ1：マシンが生成した正しいパッチは，開発者が提供したものとどのような違いがあるか.

SLDM(33，18.6%)

TABLE III. DETAILS OF DIFFERENCES BETWEEN SLDM PATCHES AND DEVELOPER-PROVIDED PATCHES

Category	Operation	#patches	Sample
Conditional	Conditional Expression Modification	5	JAID-Lang#33
	Conditional Expression Expansion	3	SimFix-Math#63
	Conditional Expression Reduction	1	Nopol-Lang#58
	Conditional Branch Removal	4	ssFix-Lang#33
	Conditional Branch Addition	8	SimFix-Closure#115
Method Call	Method Call Addition	3	Elixir-Time#15
	Method Call Replacement	4	CapGen-Lang#57
Assignment	Assignment Modification	1	ssFix-Math#80
Variable	Variable Modification	1	JAID-Chart#24
Statement	Incomplete Statement Removal	3	JAID-Closure#40

```
730     //start of patch
731     if (cArg==null&&NodeUtil.canBeSideEffected(cArg)) {
732             return CanInlineResult.NO;
733     }
734     //end of patch
```

Fig. 1. The patch for Closure#115 generated by SimFix.

図1はSimFixによって生成されたパッチである.これに対して開発者は731-733行目を直接削除している.修正の内容は異なるが，cArgがnullだとそもそも731行目まで実行できないので732行目の操作が実行されることはないため，意味的には同等の修正となる.

結果

RQ1：マシンが生成した正しいパッチは，開発者が提供したものとどのような違いがあるか.

DLSM(2，1.1%)

- 図2はCapGenによって生成されたパッチである. 開発者のパッチは7行目と8行目が入れ替わっただけのものであり，この順序はプログラムに影響しないので意味的に同等であると言える.

```
5          public Complex add(Complex rhs)
6                  throws NullArgumentException {
7   +              if ((isNaN) || (rhs.isNAN)) {    return org.apache.commons.math.complex.Complex.NaN;}
8                  MathUtils.checkNotNull(rhs);
9                  return createComplex(real + rhs.getReal(),
10                         imaginary + rhs.getImaginary());
11         }
```

Fig. 2. The patch for Math#53 generated by CapGen.

結果

RQ1：マシンが生成した正しいパッチは，開発者が提供したものとどのような違いがあるか.

DLDM(10，5.6%)

```
1056      TimeSeries copy = (TimeSeries) super.clone();
1057    + copy.minY = Double.NaN;
1058    + copy.maxY = Double.NaN;
1059      copy.data = new java.util.ArrayList();
```

a) Developer-provided patch

```
//start of patch
findBoundsByIteration();
//end of patch
if (added) {
        updateBoundsForAddedItem(item);
        //check if this addition will exceed the maximum item count...
        if (getItemCount() > this.maximumItemCount) {
                TimeSeriesDataItem d = (TimeSeriesDataItem) this.data.remove(0);
                updateBoundsForRemovedItem(d);
        }
```

b) Machine-generated patch

```
private void findBoundsByIteration() {
        this.minY = Double.NaN;
        this.maxY = Double.NaN;
        Iterator iterator = this.data.iterator();
        while (iterator.hasNext()) {
                TimeSeriesDataItem item = (TimeSeriesDataItem) iterator.next();
                updateBoundsForAddedItem(item);
        }
}
```

c) The function named *findBoundsByIteration*

Fig. 3. Patches for Chart#3.

図3はSimFixが生成したパッチと開発者が生成したパッチを示している. 開発者のパッチはcreateCopyという関数内での変更でcopyという名前のオブジェクトに2つの属性を割り当てている. この次にadd関数が呼び出されている. マシンのパッチはadd関数内で(c)に示されている別の関数を呼び出し，開発者と同じようにオブジェクトに2つの属性を割り当てている.

結果

RQ1：マシンが生成した正しいパッチは，開発者が提供したものとどのような違いがあるか.

- パッチの74.6%がSLSMに属していた. つまりマシンが生成した正しいパッチの大部分が開発者が提供するものと同じであることを示している. これは近年開発されたAPR技術の修正能力の大きな進歩を表している.

結果

RQ2：それぞれのタイプのパッチはどのように分布しているか.

TABLE VII. PATCH DISTRIBUTION FROM DEFECTS4J PROJECTS PERSPECTIVE

Project	#SLSM	#SLDM	#DLSM	#DLDM	Total
Chart	22	1	1	3	27
Closure	17	6	0	0	23
Lang	24	12	0	6	42
Math	66	9	1	1	77
Mockito	0	2	0	0	2
Time	3	3	0	0	6
Total	132	33	2	10	177

TABLE VIII. PATCH DISTRIBUTION FROM APR TECHNIQUES PERSPECTIVE

Technique	#SLSM	#SLDM	#DLSM	#DLDM	Total
CapGen	22	2	2	0	26
SimFix	23	6	0	4	33
AVATAR	18	8	0	0	26
Nopol	0	1	0	3	4
jGenProg	4	0	0	0	4
jKali	1	0	0	0	1
JAID	14	9	0	2	25
Elixir	22	4	0	0	26
ACS	16	0	0	1	17
ssFix	12	3	0	0	15
Total	132	33	2	10	177

結果

RQ3：それぞれのタイプのパッチの特徴について.

- パッチの複雑さの指標として，パッチサイズ(変更されたコードの行数)，チャンク数，変更されたファイルの数とメソッドの数を調査する.
- テストの妥当性を示すために，ステートメントカバレッジ(実行された行数)とブランチカバレッジ(実行されたブランチの数)を調査する.

結果

RQ3：APRツールが特定の特性を持つバグに対して生成するパッチは開発者が提供するものとどのような違いがあるか.

TABLE IX. RESULTS OF THE SIGNIFICANT DIFFERENCE TEST

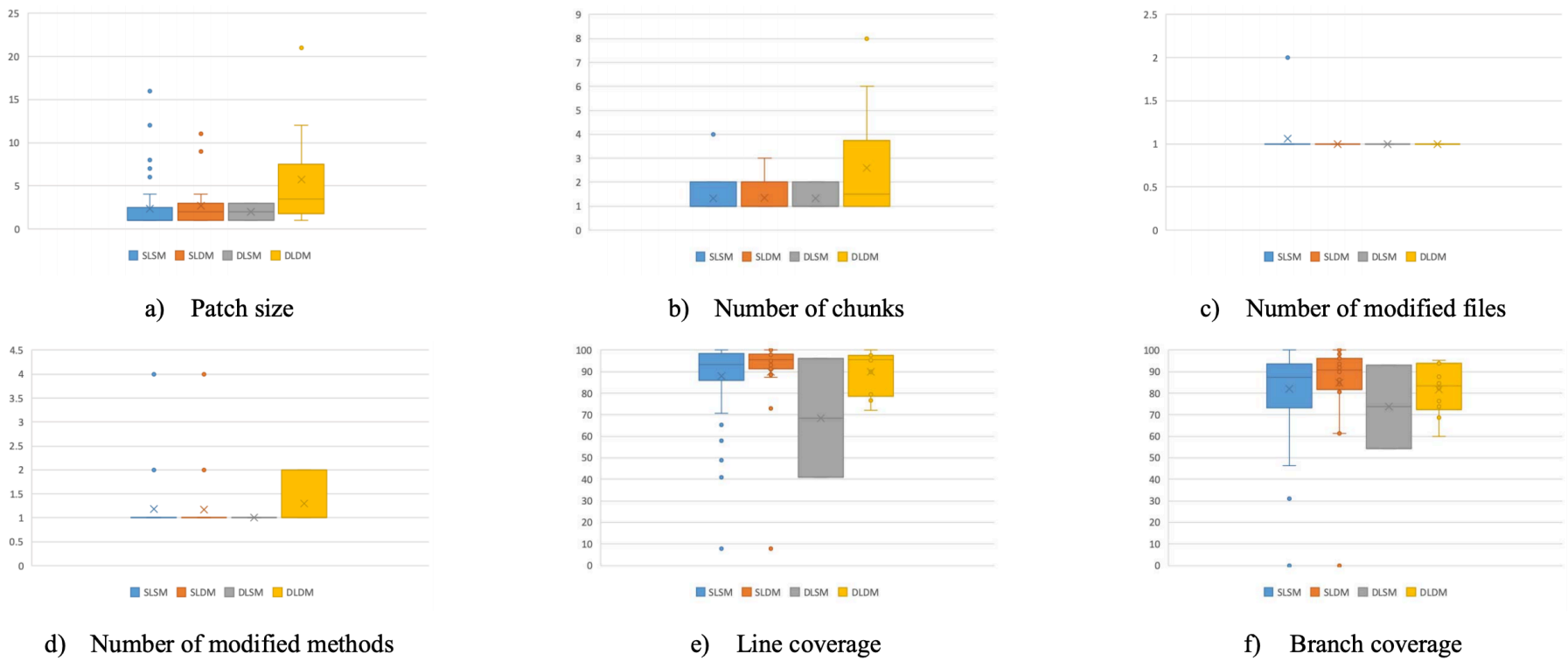
Characteristic	SLSM	SLDM		DLSM		DLDM	
	ave	ave	p-v	ave	p-v	ave	p-v
patch size	2.32	2.70	0.183	2	-	5.7	0.008
number of chunks	1.32	1.35	0.862	1	-	2.6	0.073
number of modified files	1.06	1	0.226	1	-	1	0.423
number of modified methods	1.18	1.17	0.463	1	-	1.3	0.275
line coverage	87.88%	89.84%	0.406	68.5%	-	90.04%	1
branch coverage	81.86%	84.69%	0.266	73.65%	-	81.62%	0.559

ほとんどのp値が0.05よりも大きいためこれらの違いは重要ではない.

DLDMパッチはサイズが大きいパッチを生成する傾向がある.

結果

RQ3：APRツールが特定の特性を持つバグに対して生成するパッチは開発者が提供するものとどのような違いがあるか.



a) Patch size

b) Number of chunks

c) Number of modified files

d) Number of modified methods

e) Line coverage

f) Branch coverage

Fig. 4. Distributions of Bug Characteristics

結論と所感

- 開発者と全く同じ変更でなくても，APR技術は正しいパッチを作成できることが分かった.

- 正しいパッチに関してのみ調査しており，APR技術が生成するパッチがどこまで正しいのか分からない.