自動生成されたパッチがコーディング規約違反に与える影響の分析

発表者：野口 優輝

背景

自動プログラム修正(APR)

- プログラムを修正するためのパッチを生成する技術
- 開発者の不具合修正における労力を減らすことができる

APRパッチは開発者が意図する機能を崩すことなくプログラムを修正することを期待

- 過去の不具合修正を元にパッチを生成

背景

実際のパッチ適用例

```
            try {
                return Class.forName(name, false, classLoader);
            } catch (ClassNotFoundException ex) {
-               return Class.forName(name, false, Thread.currentThread().getContextClassLoader());
+               try {
+                   return Class.forName(name, false, Thread.currentThread().getContextClassLoader());
+               } catch (ClassNotFoundException cnfe) {
+                   Class<?> cls = primitiveTypes.get(name);
+                   if (cls != null)
+                       return cls;
+                   else
+                       throw cnfe;
+               }
```

パッチが適用されたコミット

if, elseの文末に括弧が抜けている

NeedBraceの規約違反が発生

不具合修正に焦点を当てており，コーディング規約を考慮したパッチが生成されていない

- プロジェクトには独自のコーディング規約が設定されている
- コードの可読性や保守性の低下の恐れがある.

コーディング規約

ソフトウェア開発における書き方の定義

- 各プロジェクトに独自のルールが設定されている
- 静的解析ツールCheckStyleやPMDにより検査される

関連研究

- Johnsonらは，静的解析ツール使用による欠陥の特定は，目視よりも容易であり，遵守に有効である．
- Panichellaらは，コードレビュー中に指摘された違反は，80%以上が修正されていないか，後に修正されている．

APRパッチがコーディング規約に与える影響を調べた研究は存在しない

動機と目的

動機

- APR技術はプログラム修正に焦点を当てているため，コーディング規約の影響については考慮されていない
- コーディング規約に遵守しない場合，保守性の低下が引き起こされる可能性がある

目的

- 自動生成されたパッチを適用することで，プロジェクトにおけるコーディング規約にどのような影響を与えるかを分析する

分析

データセット

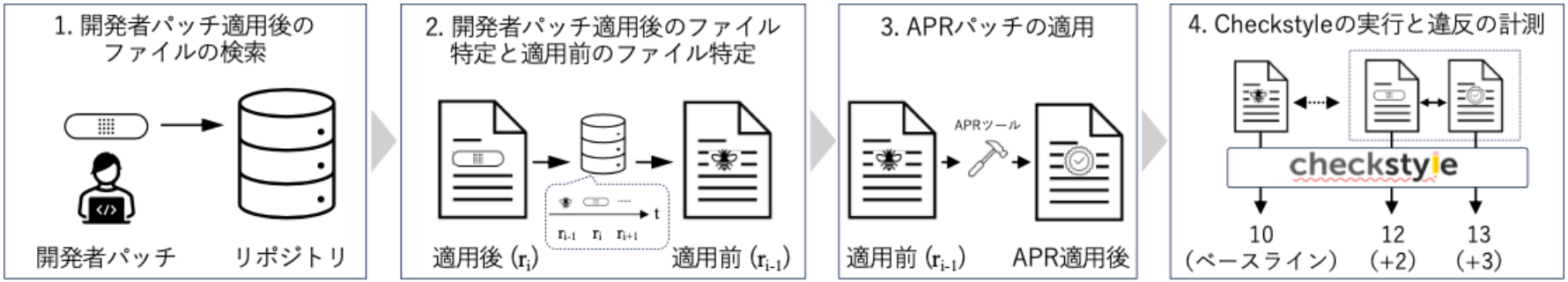
- CheckStyleを用いるLang，Math，Timeプロジェクトを対象
- 32個のAPRツールが生成したパッチ，開発者が作成したパッチ

パッチ生成元	Lang	Math	Time	パッチ生成元	Lang	Math	Time
開発者 [12]	108	193	54	jGenProg2015 [5]	0	4	0
3sFix [4]	0	2	0	jKali [23]	2	1	0
ACS [36]	3	17	1	jMutRepair [23]	0	2	0
ARJA [39]	6	11	1	KaliA [39]	0	1	0
CapGen [33]	7	14	0	kPAR [18]	1	4	0
Cardumen [24]	0	1	0	LSRepair [20]	4	1	0
CoCoNut [22]	7	19	1	Nopol2015 [5]	3	0	0
ConFix [14]	6	9	1	PraPR [7]	5	10	3
DeepRepair [34]	4	2	0	RSRepairA [39]	0	0	0
DynaMoth [6]	0	1	0	SequenceR [3]	2	8	0
ELIXIR [27]	8	12	2	SimFix [10]	9	17	1
FixMiner [15]	0	8	0	SketchFix [8]	2	6	0
GenPat [9]	4	4	0	SOFix [21]	3	13	1
GenProgA [39]	0	0	0	ssFix [35]	5	7	0
HDRepair [16]	1	5	1	Tbar [19]	7	8	0
Hercules [28]	14	31	9	Vfix [37]	7	2	0
JAID [2]	13	11	0	APR パッチ合計	123	241	21
				All（開発者＋ APR）	231	434	75

分析方法

分析の流れ

- 開発者パッチ適用後のファイルの検索
- 開発者パッチ適用前後のファイルの特定
- APRパッチの適用
- CheckStyleの実行と規約違反の計測と比較



分析方法の図

分析の流れ ①

開発者パッチ適用前後のファイル検索・特定

–開発者パッチを利用し，類似するコミットを検索

```
@@ -236,6 +236,8 @@
     * class here is a workaround, see the JIRA issue LANG-626. </p>
     */
     static class ClassLoaderAwareObjectInputStream extends ObjectInputStream {
+        private static final Map<String, Class<?>> primitiveTypes =
+                new HashMap<String, Class<?>>();
         private ClassLoader classLoader;

         /**
```

開発者パッチ

+を含むコミットを検索

```
234  236      * class here is a workaround, see the JIRA issue LANG-626. </p>
235  237 +    */
236  238      static class ClassLoaderAwareObjectInputStream extends ObjectInputStream {
     239 +       private static final Map<String, Class<?>> primitiveTypes =
     240 +               new HashMap<String, Class<?>>();
237  241         private ClassLoader classLoader;
```

類似するコミット

直前のリビジョンから
パッチ適用前のjava
ファイルを取得

分析の流れ ②

APRパッチの適用

- パッチ適用前のjavaファイルにAPRツールごとの適用

CheckStyleの適用

- 各プロジェクトが定義したファイル(projectスタイル)
- GoogleとSunが提供するファイル(Googleスタイル，Sunスタイル)

CheckStyleの計測

- パッチ適用前後でのコーディング規約違反数の変化を分析

結果

APRパッチはおよそ40%以上のコーディング規約違反数を増加させることがわかった.

- プロジェクトのコード品質を悪化させる可能性がある

プロジェクト	Style	増加 (%)		減少 (%)		変化なし (%)	
		APR	開発者	APR	開発者	APR	開発者
Lang	project	**49.2(60)**	10.5(11)	3.3(4)	**5.7(6)**	47.5(58)	**83.8(88)**
	Google	**60.7(74)**	55.2(58)	4.1(5)	**11.4(12)**	**35.2(43)**	33.3(35)
	Sun	**56.6(69)**	34.3(36)	3.3(4)	**8.6(9)**	40.2(49)	**57.1(60)**
Math	project	**47.3(104)**	13.6(25)	3.2(7)	**15.8(29)**	49.5(109)	**70.7(130)**
	Google	**43.2(95)**	41.8(77)	5.9(13)	**15.8(29)**	**50.9(112)**	15.2(28)
	Sun	**43.6(96)**	23.9(44)	5.5(12)	**12.5(23)**	50.9(112)	**63.6(117)**
Time	project	**20.0(4)**	1.9(1)	**0.0(0)**	**0.0(0)**	80.0(16)	**98.1(52)**
	Google	40.0(8)	**69.8(37)**	0.0(0)	**5.6(3)**	**60.0(12)**	24.5(13)
	Sun	**40.0(8)**	32.1(17)	0.0(0)	**5.6(3)**	60.0(12)	**62.7(33)**

議論

APRパッチと開発者パッチの規約違反数

- APRパッチと開発者パッチで規約違反数は増加している
  - APRパッチ：全てのスタイルで増加
  - 開発者パッチ：Googleスタイル，Sunスタイルで40%近く

コーディング規約違反の違いを分析(Google，Sun)

- 各パッチ適用で発生した規約違反を集計
- 違反ごとにパッチの割合を計算

APRパッチと開発者パッチでどのように規約違反が違うか

議論｜Googleスタイル

- APRパッチではスペースの扱いに関する違反が多く，インデントに関する違反はAPR，開発者のどちらも多かった
  - Googleスタイルは2つの半角スペースにしている
  - 開発者パッチはインデント以外の違反はあまりなかった

APRパッチ		開発者パッチ	
Indentation	55.2%	Indentation	95.9%
WhitespaceAround	50.0%	LineLength	11.1%
WhitespaceAfter	47.1%	WhitespaceAround	8.2%
FiletabCharacter	40.2%	OperatorWrap	6.4%
ParenPad	13.8%	RequireEmptyline-BeforeBlockTabGroup	4.7%

各パッチ上位5件のコーディング規約違反

議論｜Sunスタイル

APRパッチではスペースの扱いに関する違反，開発者パッチではコードの長さや，マジックナンバーの違反が多かった

- コードの長さ：1行あたりのコード数制限
- マジックナンバー：コードで急に現れる具体的な数字

APRパッチ		開発者パッチ	
WhitespaceAround	50.9%	LineLength	53.9%
WhitespaceAfter	48.0%	MagicNumber	38.2%
FiletabCharacter	41.5%	WhitespaceAround	12.7%
LineLength	22.2%	RegexpSingleline	11.8%
ParenPad	14.6%	WhitespaceAfter	11.8%

各パッチ上位5件のコーディング規約違反

結果・議論のまとめ

APRパッチでは特にスペースに関する違反が多かった

- APR技術はコードのトークン化や抽象構文木化し，利用されることが一般的なためだと考えられる

APR研究の今後の課題

- コードの可読性に影響する規約違反もあり，開発者の意図に反して保守性を低下させる恐れがある

プログラムの保守性を低下させる要因は可能な限り，排除するべき

追加分析

違反はIDE(統合開発環境)で自動修正可能では？

- スペースに関する違反や，括弧の位置などの違反はIDEで自動修正することができる

パッチ適用から，違反修正コミットが存在するか検証

1. 開発者パッチ適用後のコミットを探索
2. コミットが存在する場合，どのくらいの日数で修正が行われているのかを検証

追加分析｜分析方法

パッチ適用部分が変更されているコミットを探索

```
        int n = getImaginaryCharacter().length();
        startIndex = pos.getIndex();
        int endIndex = startIndex + n;
-       if (source.substring(startIndex, endIndex).compareTo(
+       if ((startIndex >= source.length()) ||
+           (endIndex > source.length()) ||
+           source.substring(startIndex, endIndex).compareTo(
            getImaginaryCharacter()) != 0) {
            // set index back to initial, error index should be the start index
            // character examined.
```

このテキストを含むコミットを探索

UnnecessaryParentheses
という違反が発生

パッチが適用されたコミット

x日後

```
        final int startIndex = pos.getIndex();
        final int endIndex = startIndex + expected.length();
-       if ((startIndex >= source.length()) ||
-           (endIndex > source.length()) ||
-           (source.substring(startIndex, endIndex).compareTo(expected) != 0)) {
+       if (startIndex >= source.length() ||
+           endIndex > source.length() ||
+           source.substring(startIndex, endIndex).compareTo(expected) != 0) {
            // set index back to start, error index should be the start index
            pos.setIndex(startIndex);
            pos.setErrorIndex(startIndex);
```

余分な括弧が削除されている

違反が修正されているコミット

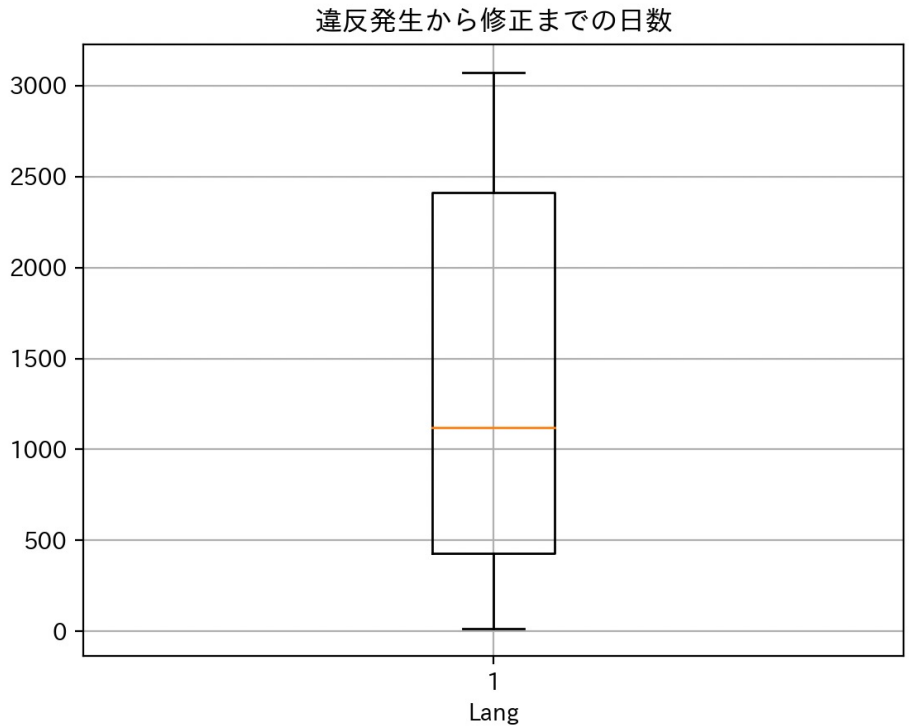
追加分析｜データ

- projectスタイルで違反が出た開発者パッチを対象
  - Timeに関しては，違反が出たパッチが1つのため除く

project	パッチの個数	違反の個数
Lang	13	13
Math	41	51

追加分析結果｜Lang

- 規約違反13個中，11個が修正・変更・削除されていた
  - 修正されていない規約違反は2個だった

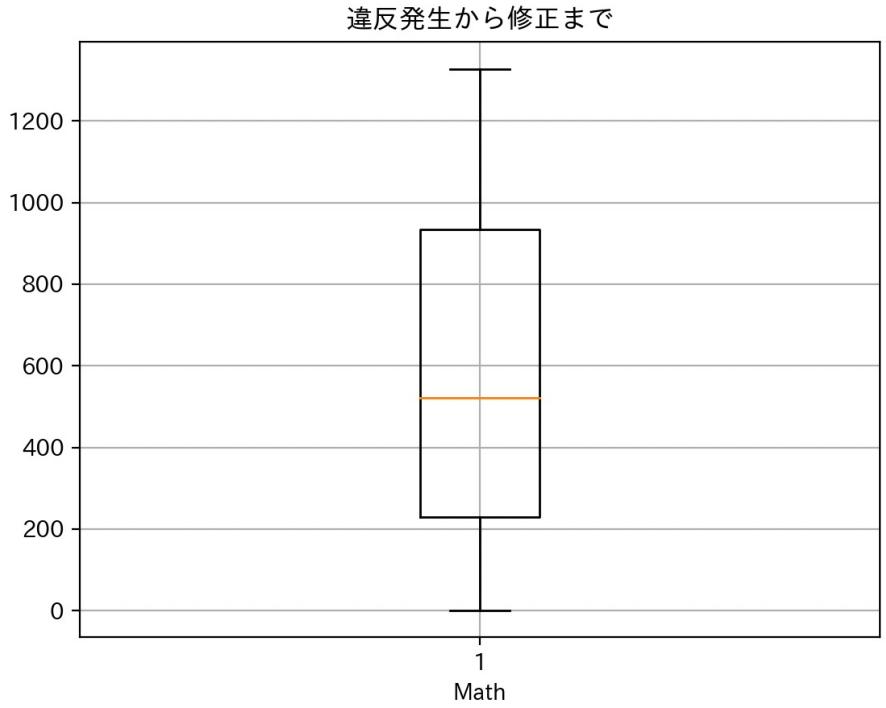


	日数(日)
平均値	1430
中央値	1120

追加分析結果｜Math

規約違反51個中，30個が修正・変更・削除されていた.

- Indentation(19個)は類似するコミットが多数のため除く
- 修正されていない規約違反は2個だった



	日数(日)
平均値	978
中央値	522

追加分析｜まとめ

- パッチ適用における規約違反は大半が修正されていた
  - Langでは違反が13個中11個
  - Mathでは違反がIndentationを除くと32個中30個

- 修正までの日数はかなりかかっていた
  - Lang，Mathそれぞれ中央値が1430日，978日と高かった.
  - コードの品質が低下する恐れ

バグの発生やメンテナンスの難しさにつながってくる

まとめ

- 自動生成されたパッチ(APRパッチ)がコーディング規約違反に与える影響を分析
  - APRパッチは適用前後で，約40%のコーディング規約違反数を増加させることがわかった.
  - APRパッチは主に，スペースの扱いに関する違反を多く出すことがわかった.

- コーディング規約違反が修正までにかなりに日数がかかる
  - Lang，Mathでかなりの日数かかっていたので，バグの発生などを防ぐために，早めに対処するべき.