

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

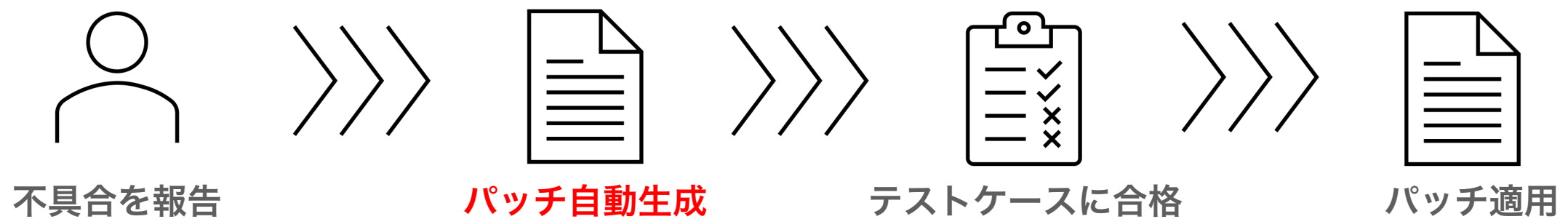
オープンソースソフトウェア工学研究室
学生番号：60256207
発表者：野口 優輝

背景

自動プログラム修正 (Automated Program Repair)

- プログラムを修正するためのパッチを自動で生成する技術

自動プログラム修正の流れ



- 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

自動生成パッチ (APRパッチ)

- APRツールによって自動生成されたパッチ
- 開発者が意図する機能を崩すことなく不具合を修正することが期待されている^[1]

[1] Durieux, T., Martinez, M., Monperrus, M., Sommerard, R., and Xuan, J.: Automatic repair of real bugs: an experience report on the defects4j dataset, Empirical Software Engineering, 2017, pp1936-1964

APRパッチの品質

パッチの正しさはテストケースに合格するかで保証

- プログラムの振る舞いにおいての正しさは保証されるが、コードの中身（書き方）までの正しさは保証されない

パッチが生成したコードの品質

- コーディング規約に従ったコードは可読性や保守性を保ち、コードの品質を高めることができる^[2]

→APRパッチもコードの品質を考慮することは重要

コーディング規約に沿ったパッチを生成する必要がある

[2]織田 泰輔, 阿萬 裕久, 佐々木 隆志, 川原 稔, オープンソース開発でのコード修正におけるコーディング規約違反の変化に関する調査, 情報処理学会研究報告会2014, pp1-8

コーディング規約

ソフトウェアにおけるコードの書き方のガイドライン

- 各プロジェクトに独自のルールが設定されている

静的解析ツールで自動検査

- JavaではCheckStyleなどでコーディング規約違反を検出^[3]

関連研究

- 静的解析ツール使用による欠陥の特定は、目視より容易であり、コーディング規約遵守に有効である^[4]

APRパッチがコーディング規約違反に
与える影響を調べた研究は存在しない

[3] Beller, M., Bholanath, R., McIntosh, S., and Zaidman, A.: Analyzing the state of static analysis: a large-scale evaluation in open source software, Proc. SANER, 2016, pp. 470–481.

[4] Panichella, S., Arnaoudova, V., Di Penta, M., and Antoniol, G.: Would static analysis tools help developers with code reviews?, Proc. SANER, 2015, pp. 161–170.

動機と目的

動機

- APRパッチは様々なプロジェクトの不具合情報[1]を元に生成
- ただし、**遵守すべき規約はプロジェクト毎に異なる**
- APRパッチの適用によりコーディング規約違反が混入すると、開発者が関知しない保守性の低下が起こる恐れがある

目的

- APRパッチがコーディング規約違反に与える影響を明らかにする

リサーチクエスチョン

RQ1. APRパッチ適用は規約違反数はどの程度増加するか

RQ2. APRパッチ適用により発生する規約違反に特徴はあるか

データセット

- [5]の内, CheckStyleを利用するプロジェクト
 - Apache Commons Lang
 - Apache Commons Math
 - Joda Time
- 開発者が作成したパッチ
- 32種類のAPRツールが生成したパッチ

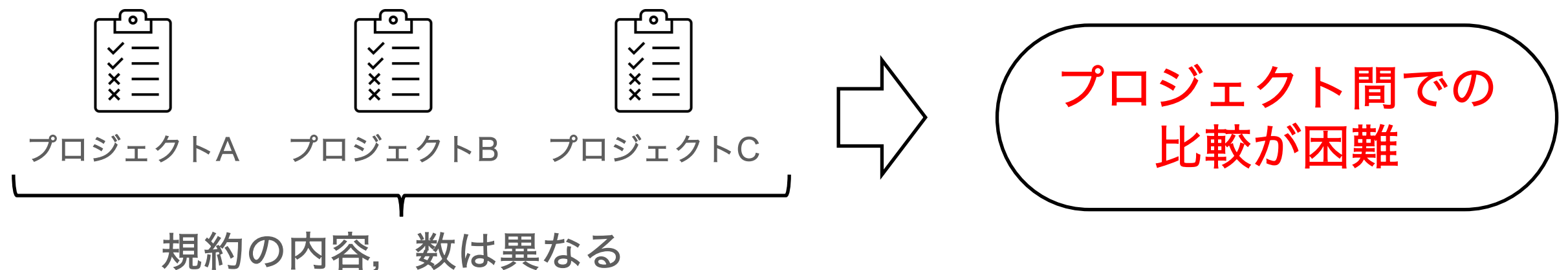
パッチ生成元	Lang	Math	Time	パッチ生成元	Lang	Math	Time
開発者 [12]	105	184	53	jGenProg2015 [5]	0	4	0
3sFix [4]	0	2	0	jKali [23]	2	1	0
ACS [36]	3	15	1	jMutRepair [23]	0	1	0
ARJA [39]	6	10	1	KaliA [39]	0	1	0
CapGen [33]	7	14	0	kPAR [18]	1	4	0
Cardumen [24]	0	1	0	LSRepair [20]	4	1	0
CoCoNut [22]	7	17	1	Nopol2015 [5]	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	16	1
FixMiner [15]	0	8	0	SketchFix [8]	2	6	0
GenPat [9]	4	3	0	SOFix [21]	3	13	1
GenProgA [39]	0	0	0	ssFix [35]	5	7	0
HDRRepair [16]	1	5	1	Tbar [19]	7	8	0
Hercules [28]	13	30	8	Vfix [37]	7	0	0
JAID [2]	13	11	0	APR パッチ合計	122	220	20
				All (開発者+ APR)	231	434	75

[5] Tian, H., Li, Y., Pian, W., Kaboré, A. K., Liu, K., Habib, A., Klein, J., and Bissyandé, T. F.: Predicting patch correctness based on the similarity of failing test cases, Trans. Softw. Eng. Methodol., Vol. 31, No. 4(2022), pp. 1-30.

使用する設定ファイル

projectスタイル

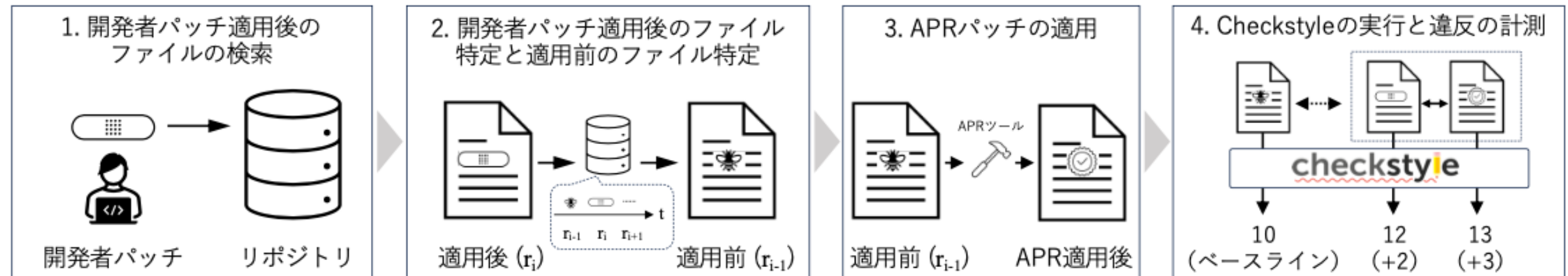
- プロジェクト独自で定義されたCheckStyleファイル
- プロジェクトごとに規約の内容や数が異なる



Googleスタイル, Sunスタイル

- 一般的に広く参照されているCheckStyleファイル
- projectスタイルを用いるよりも, 規約違反数の変化をよく捉えることができる

RQ1 | 分析手順



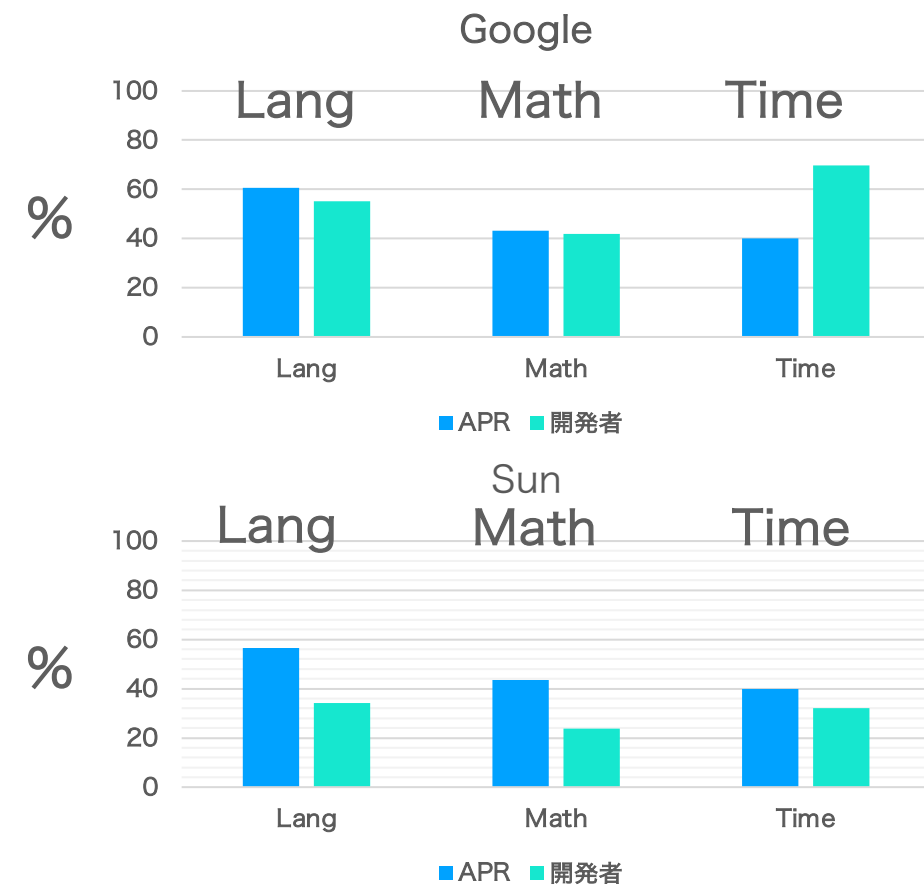
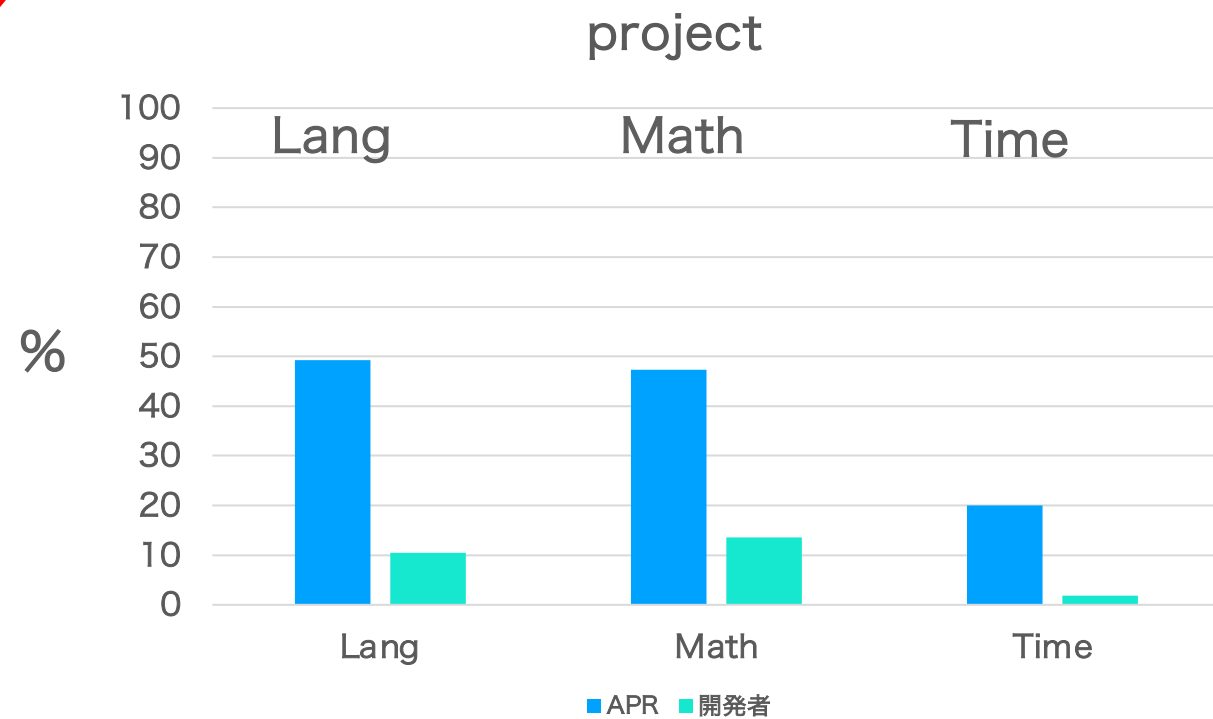
1. 開発者パッチ適用後のファイルを検索し，コミットを特定
2. 開発者パッチ適用前後のファイルを特定
3. APRツールごとにAPRパッチの適用
4. CheckStyle実行とコーディング規約違反の計測と比較

計算方法

$$\text{割合(\%)} = \frac{\text{増加したパッチの個数}}{\text{プロジェクトにおけるパッチの個数}} \times 100$$

(増加)

RQ1 | 分析結果



全てのスタイルで，APRパッチは規約違反数を増加させる割合が高い

- projectスタイルでは，開発者パッチと比較しても，特に割合が高くなっていることがわかる

RQ2：分析方法

Google, Sunスタイルを用いて分析

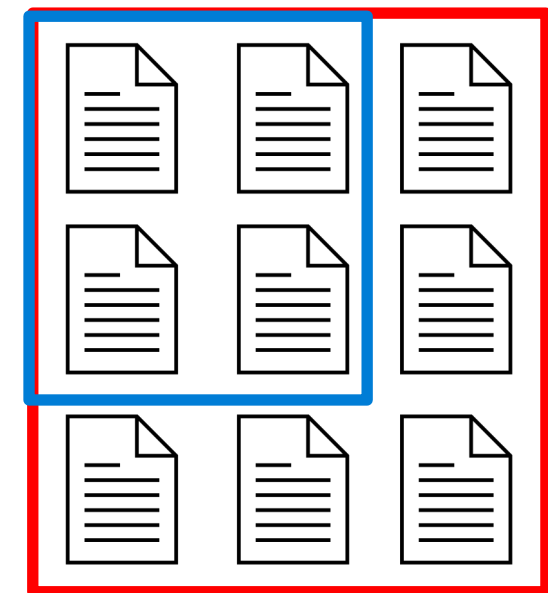
- projectスタイルでは、プロジェクトごとにルールが異なる
→APRが発生させやすい違反の特徴が定まらない

分析手順

- パッチ適用で発生した規約違反を集計
- 規約違反ごとにパッチの割合を計算

計算方法

$$\text{割合} = \frac{\text{ある違反が発生したパッチの総数}}{\text{違反数が増加したパッチの総数}} \times 100$$

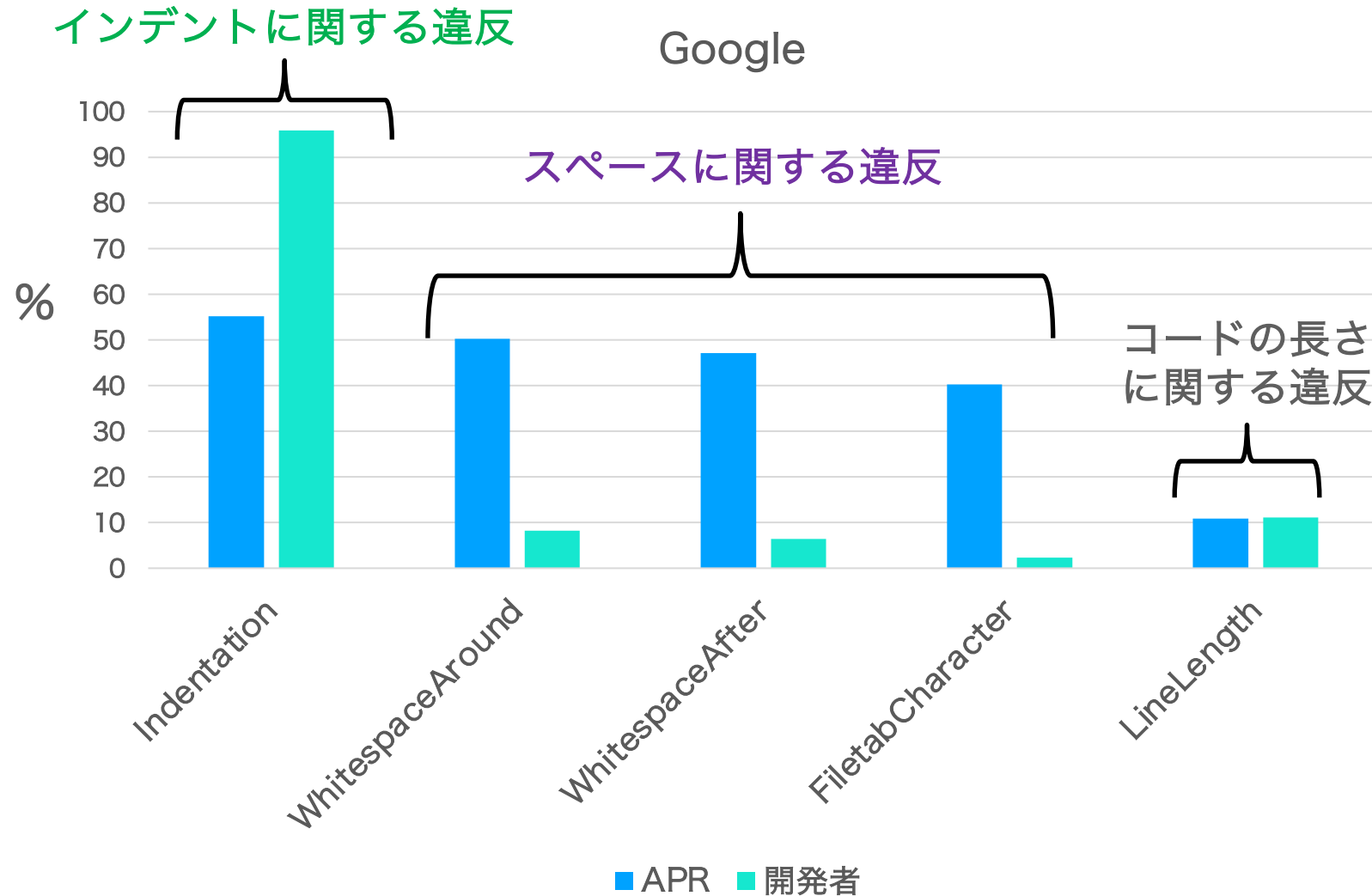


違反Aが発生

$$\frac{4}{9} \times 100 = 44.44...$$

約44%, 違反Aが発生している

RQ2 | 分析結果 (Google)

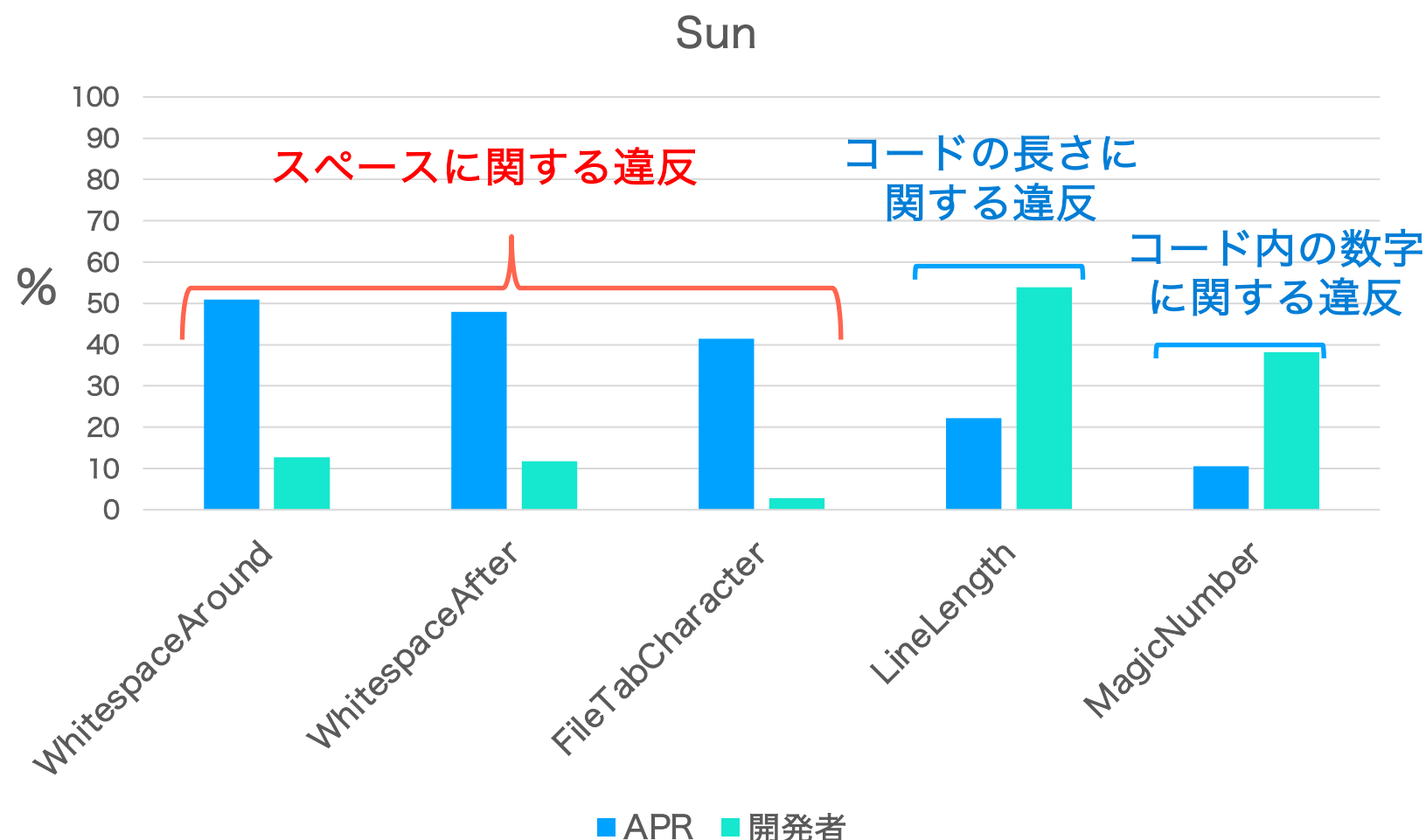


違反名	説明
Indentation	インデントがルールに合致
Whitespace Around	トークンの前後に空白があるか
Whitespace After	特定の文字の後に空白があるか
FileTab Character	タブ文字ではなく空白を使ってるか
LineLength	1行あたりにある文字数

APRは開発者と同様にインデントの違反が多い

APRはスペースに関する違反が多い

RQ2 | 分析結果 (Sun)



違反名	説明
Whitespace Around	トークンの前後に空白があるか
Whitespace After	特定の文字の後に空白があるか
FileTab Character	タブ文字ではなく空白を使ってるか
LineLength	1行あたりにある文字数
MagicNumber	マジックナンバーを使用しているか

Google同様, APRはスペースに関する違反が多い

APRは開発者と比較して, コードの長さ, マジックナンバーに関する違反は少ない

考察

RQ1

- projectスタイルでは、APRパッチは開発者パッチよりも規約違反数を増加させる割合が高くなっている。
 - 開発者はプロジェクトの規約を意識してパッチを作成している
 - APRはプロジェクトの規約に準拠したパッチを生成していない

RQ2

- APRパッチはGoogle, Sunスタイルともにスペースに関する違反を多く発生させていた
 - APR技術はコードのトークン化や抽象構文木化を利用していることで、コードのスタイルが失われやすい

まとめ

目的

- APRパッチがコーディング規約違反に与える影響を明らかにする

結果

- APRパッチ適用は規約違反数を増加させる割合が高い
- APRパッチはスペースに関する違反を発生させやすい

今後の展望

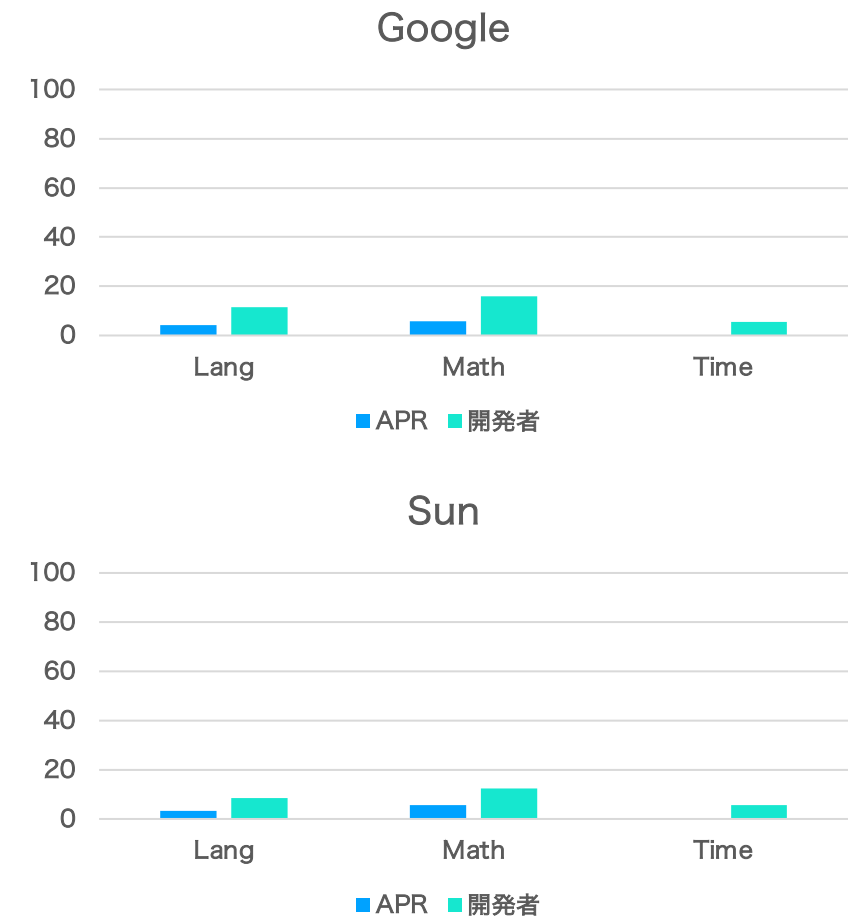
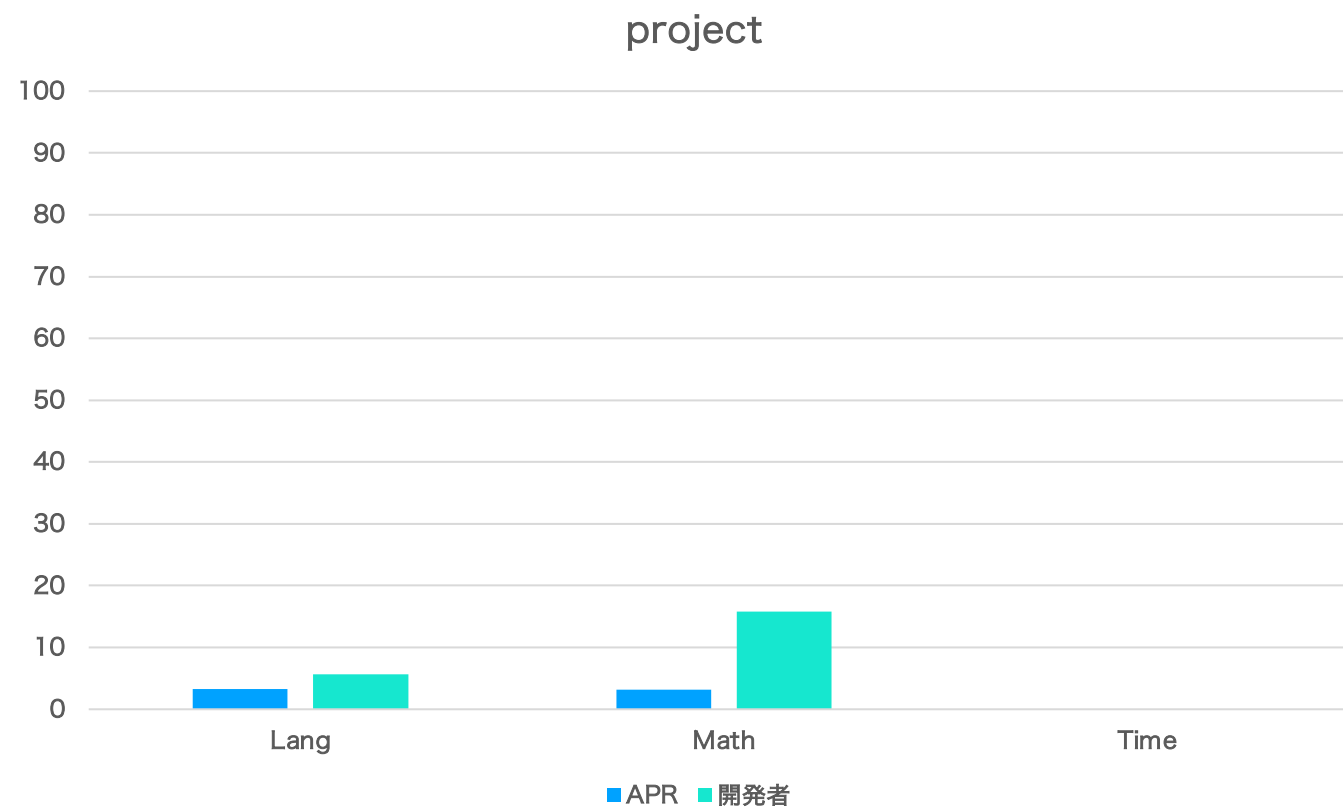
- APRツールごとに、コーディング規約違反数の変化や、発生する規約違反の特徴を比較

発表状況

| 野口 優輝, 木村 祐太, 伊原 彰紀, 大平 雅雄, "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", FOSE2023 第30回ソフトウェア工学の基礎ワークショップ, pp.83-92, 2023年11月

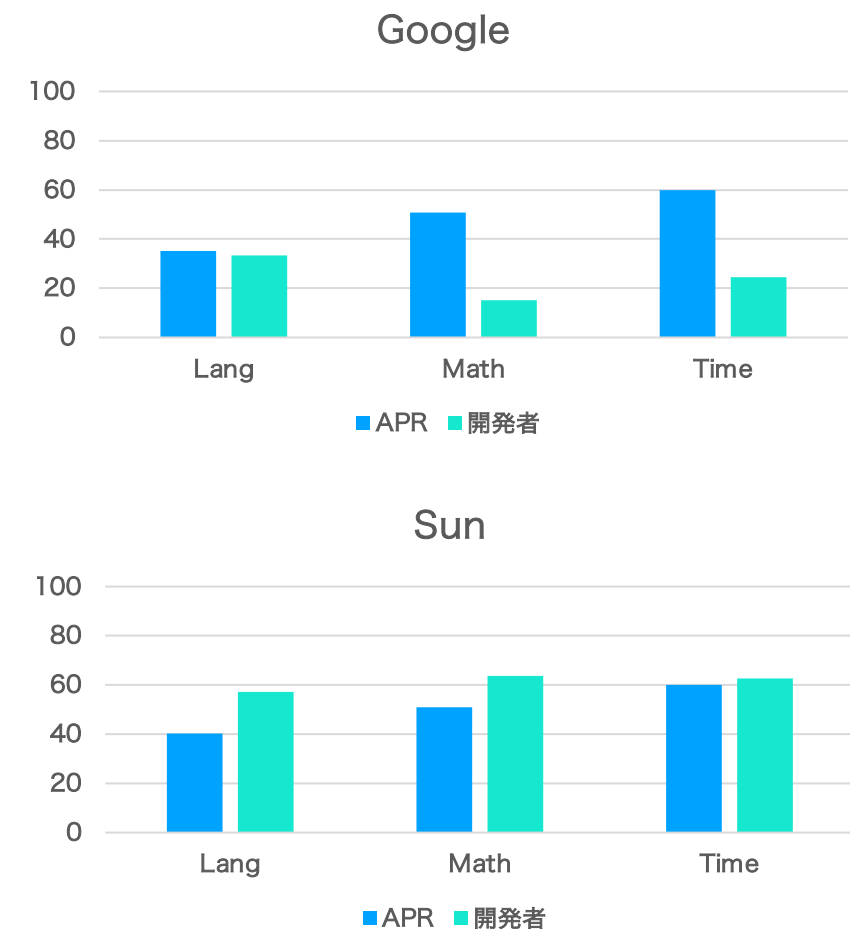
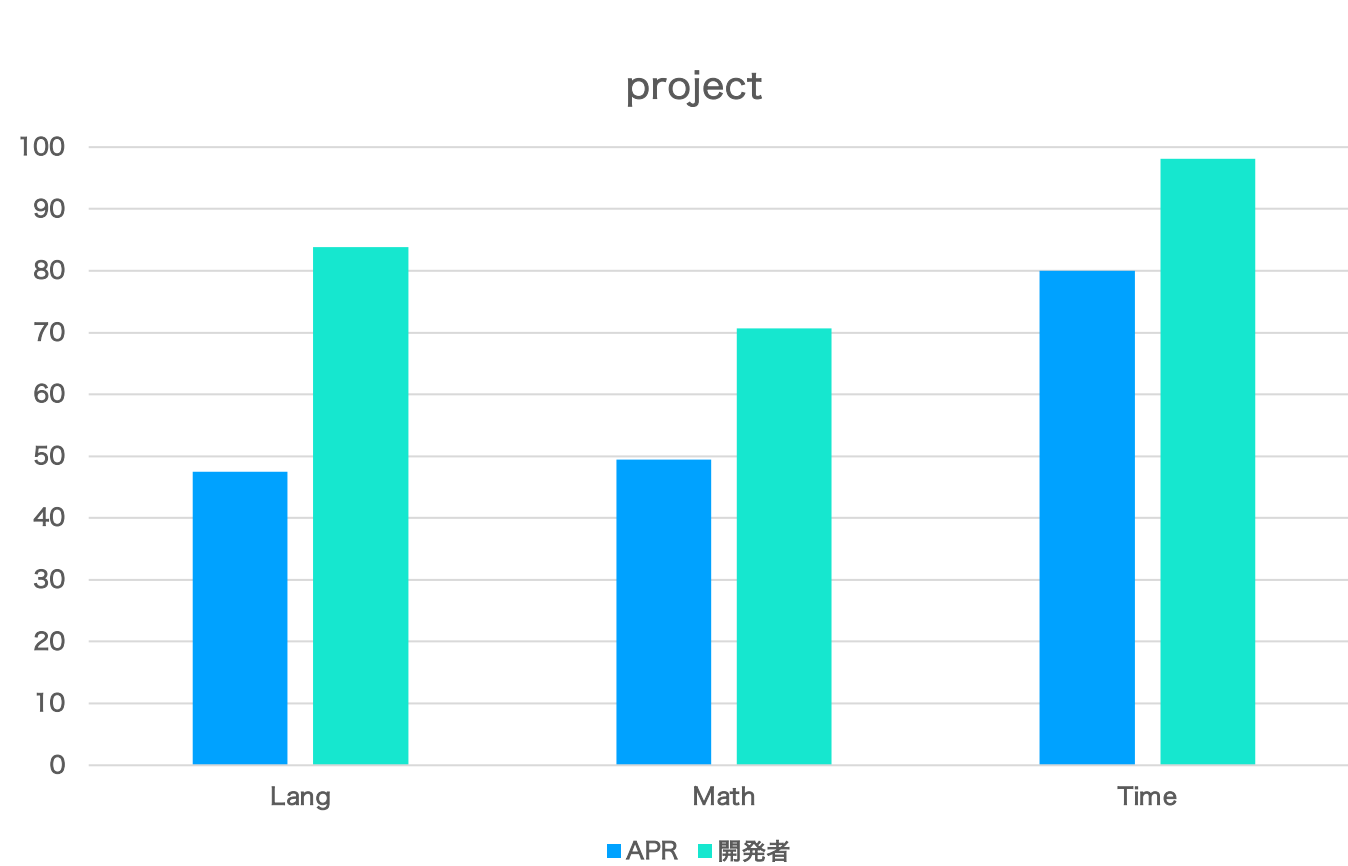
補足資料

規約違反数が減少したパッチの割合



APR, 開発者ともに, 規約違反数を減少させるパッチの割合は少なかった

規約違反数が変化しないパッチの割合



APR, 開発者ともに、半数以上のパッチは規約違反数が変化しなかった

APRが発生させる違反（その他）

Google

規約違反名	説明
ParenPad	括弧内の空白が正しいか
LeftCurly	左中括弧の出現位置が正しいか
NoWhitespace Before	各トークンの前に空白がないか
NeedBraces	中括弧が存在するか
RequireEmptyLine BeforeBlockTagGroup	Javadocがある場合、前に1行空白があるか

Sun

規約違反名	説明
ParenPad	括弧内の空白が正しいか
NoWhitespace Before	各トークンの前に空白がないか
SimplifyBoolean Expression	冗長なboolean演算子があるか
LeftCurly	左中括弧の出現位置が正しいか
NeedBraces	中括弧が存在するか