

Mining Software Defects: Should We Consider Affected Releases?

Suraj Yatish^{}, Jirayus Jiarpakdee^{}, Patanamon Thongtanunam^{}, Chakkrit Tantithamthavorn^{}
^{} The University of Adelaide. ^{} Monash University. ^{} The University of Melbourne.
Email: suraj.yatish@adelaide.edu.au, jirayus.jiarpakdee@monash.edu,
patanamon.thongtanunam@unimelb.edu.au, chakkrit.tantithamthavorn@monash.edu

ICSE '19 Proceedings of the 41st International Conference on Software Engineering Pages 654-665

簡単にまとめると

- Bug Predictionなどの分野で用いられるDefect Datasetに問題がありそうなのでデータセットの作り方について調査してみた

背景

- Defect Datasetsはソフトウェア工学の研究において広く理由されている
- Defect モジュール予測では、過去の欠陥を学習することで予測モデルを構築している
- しかし、Datasetsの作り方によってDatasetsの性質が変わる可能性がある

Defect Datasetの作成方法

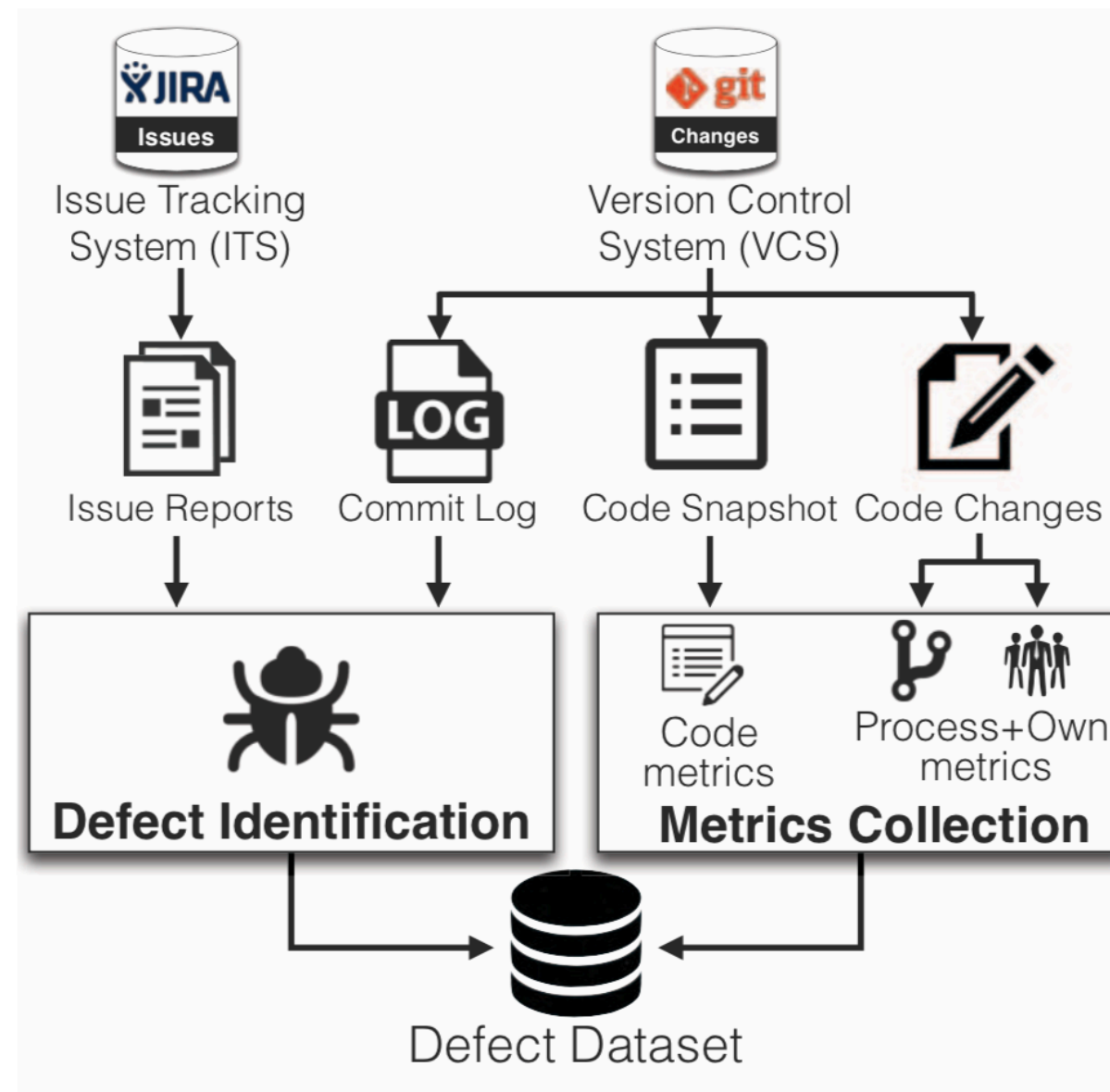
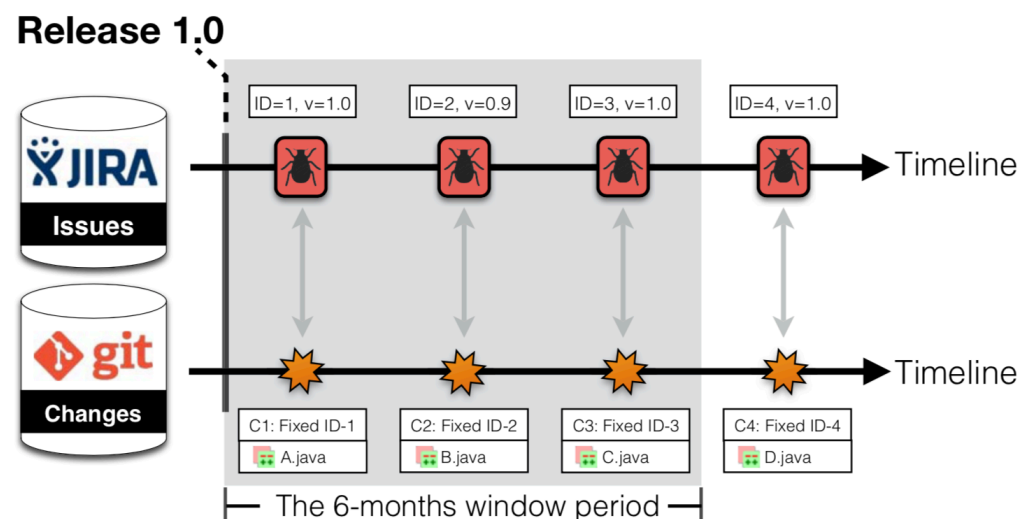


Fig. 1: An overview of defect data preparation steps.

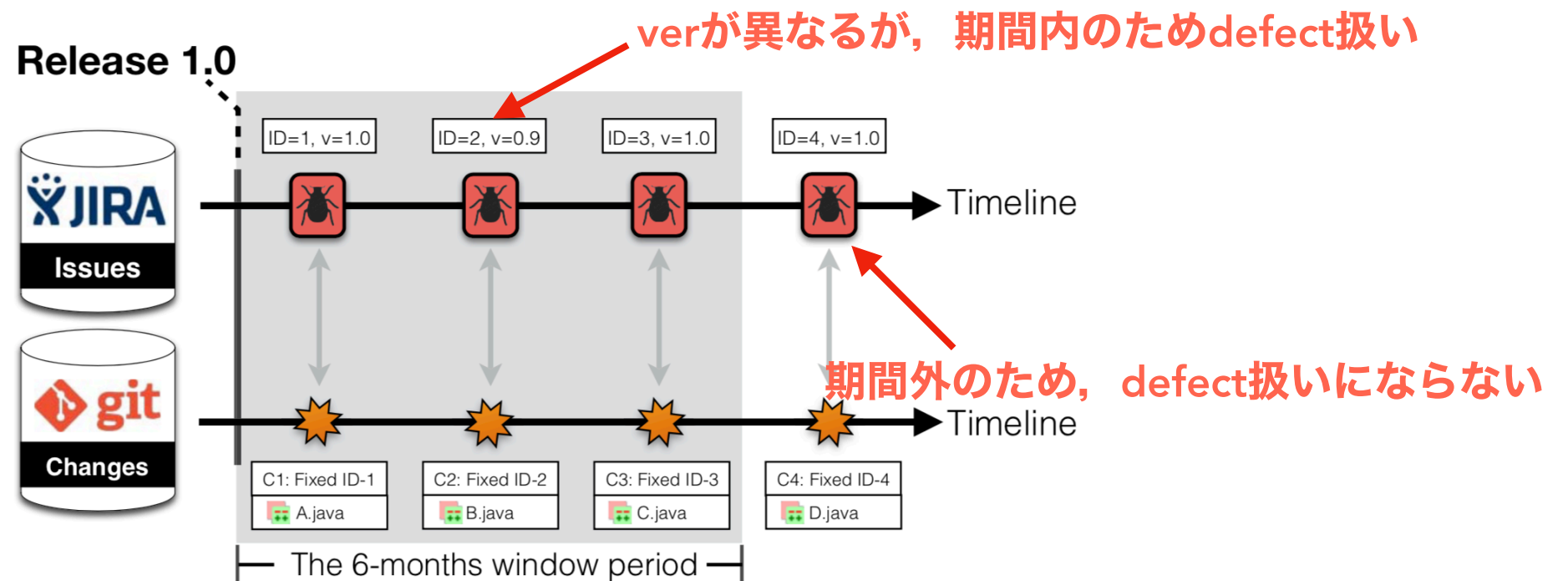
Dataset作成アプローチ

- Heuristic Approach
 - 正規表現 (e.g., fix(e[ds])?, bugs?, defects?) を用いて不具合表と修正コミットを紐づけ
- Realistic Approach
 - JIRAなどのツールでは、不具合表から修正コミットへの追跡リンクが存在するためリンクを利用して特定



Heuristic Approachの課題

- 特定のリリース後の考慮するデータ期間内の不具合修正コミットは、すべてリリースに影響していると仮定している
- リリースに関する不具合修正は、すべてリリース後の考慮するデータ期間内に修正されると仮定している
- Heuristic approachでは、データセットにノイズを発生させる可能性がある



目的

- Heuristic ApproachとRealistic Approachで作成される Datasetsの性質の違いと、 Datasetsによる予測モデルへの影響の調査

リサーチクエスト

1. Heuristic defects datasetsとRealistic defects datasetsは異なるか？
2. 各Datasetsによって、予測モデルの精度への影響はあるか？
3. 各Datasetsで予測モデルを作成した場合、欠陥を引き起こすモジュールの予測ランクへの影響はあるか？

対象プロジェクト

- 不具合報告，修正済み欠陥が複数あるプロジェクト
- 不具合報告と修正コミットのリンクが追跡可能であるプロジェクト（JIRAを使っている）

TABLE I: A statistical summary of the studied systems.

Name	Description	#DefectReports	No. of Files	Defective Rate	KLOC	Studied Releases
ActiveMQ	Messaging and Integration Patterns server	3,157	1,884-3,420	6%-15%	142-299	5.0.0, 5.1.0, 5.2.0, 5.3.0, 5.8.0
Camel	Enterprise Integration Framework	2,312	1,515-8,846	2%-18%	75-383	1.4.0, 2.9.0, 2.10.0, 2.11.0
Derby	Relational Database	3,731	1,963-2,705	14%-33%	412-533	10.2.1.6, 10.3.1.4, 10.5.1.1
Groovy	Java-syntax-compatible OOP for JAVA	3,943	757-884	3%-8%	74-90	1.5.7, 1.6.0.Beta_1, 1.6.0.Beta_2
HBase	Distributed Scalable Data Store	5,360	1,059-1,834	20%-26%	246-534	0.94.0, 0.95.0, 0.95.2
Hive	Data Warehouse System for Hadoop	3,306	1,416-2,662	8%-19%	287-563	0.9.0, 0.10.0, 0.12.0
JRuby	Ruby Programming Lang for JVM	5,475	731-1,614	5%-18%	105-238	1.1, 1.4, 1.5, 1.7
Lucene	Text Search Engine Library	2,316	8,05-2,806	3%-24%	101-342	2.3.0, 2.9.0, 3.0.0, 3.1.0
Wicket	Web Application Framework	3,327	1,672-2,578	4%-7%	109-165	1.3.0.beta1, 1.3.0.beta2, 1.5.3

分析方法の概要

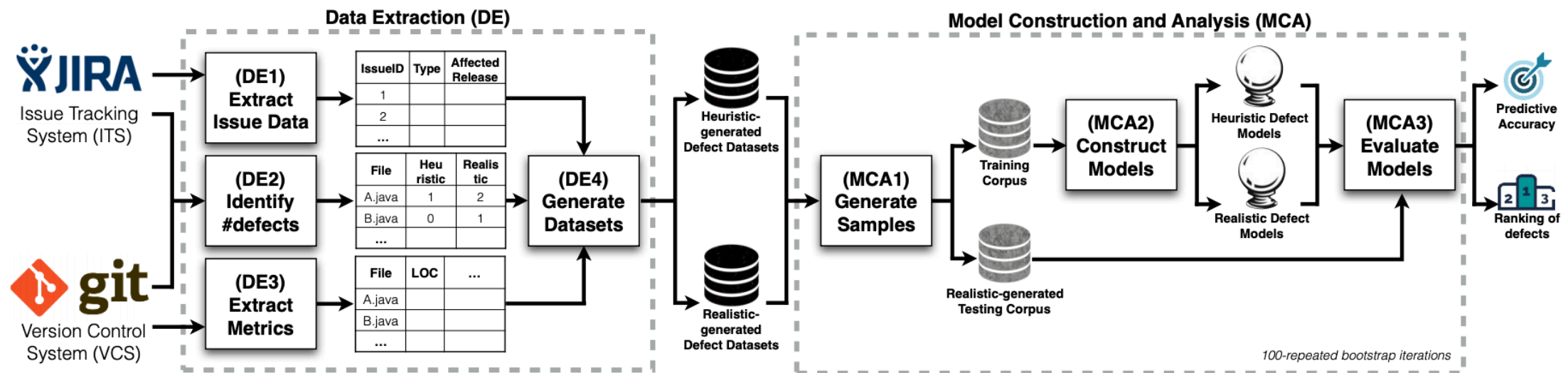


Fig. 3: An overview of our case study design.

(DE1) : JIRAから不具合表を抽出 (ID, Ver, 種類など)

(DE2) : 各Approachで欠陥を検出

(DE3) : メトリクスの抽出 (Code Metrics, Process Metrics, Ownership Metrics)

(DE4) 各リリース後のDefect count datasets (各モジュールに含まれる欠陥数) と
Binary defect datasets (各モジュールに欠陥が含まれるか) を作成する

(MA1) モデル学習データとRealistic Approachで作成したテストデータに分割

(MA2) Heuristic datasetsとRealistic Datasetsでモデルを作成

(MA3) 各モデルを評価

抽出するメトリクス

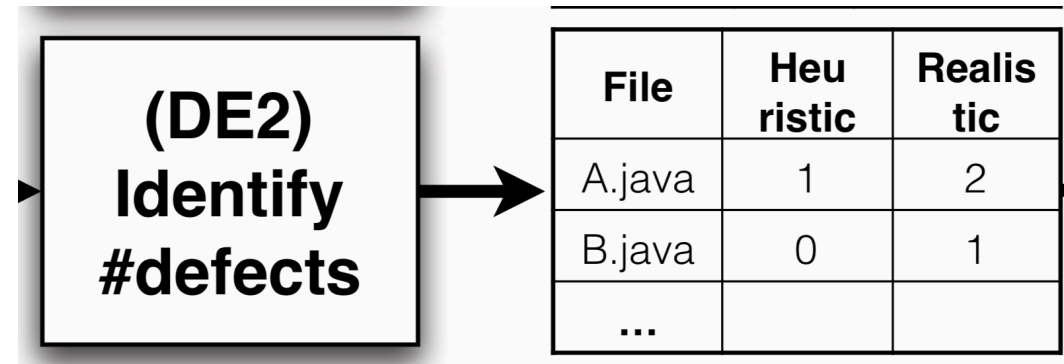
TABLE II: A summary of the studied code metrics.

Granularity	Metrics	Count
File	AvgCyclomatic, AvgCyclomaticModified, AvgCyclomaticStrict, AvgEssential, AvgLine, AvgLineBlank, AvgLineCode, AvgLineComment, CountDeclClass, CountDeclClassMethod, CountDeclClassVariable, CountDeclFunction, CountDeclInstanceMethod, CountDeclInstanceVariable, CountDeclMethod, CountDeclMethodDefault, CountDeclMethodPrivate, CountDeclMethodProtected, CountDeclMethodPublic, CountLine, CountLineBlank, CountLineCode, CountLineCodeDecl, CountLineCodeExe, CountLineComment, CountSemicolon, CountStmt, CountStmtDecl, CountStmtExe, MaxCyclomatic, MaxCyclomaticModified, MaxCyclomaticStrict, RatioCommentToCode, SumCyclomatic, SumCyclomaticModified, SumCyclomaticStrict, SumEssential	37
Class	CountClassBase, CountClassCoupled, CountClassDerived, MaxInheritanceTree, PercentLackOfCohesion	5
Method	CountInput_{Min, Mean, Max}, CountOutput_{Min, Mean, Max}, CountPath_{Min, Mean, Max}, MaxNesting_{Min, Mean, Max}	12

TABLE III: A summary of the studied process and ownership metrics.

Metrics	Description
<i>Process Metrics</i>	
COMM	The number of Git commits
ADDED_LINES	The normalized number of lines added to the module
DEL_LINES	The normalized number of lines deleted from the module
ADEV	The number of active developers
DDEV	The number of distinct developers
<i>Ownership Metrics</i>	
MINOR_COMMIT	The number of unique developers who have contributed less than 5% of the total code changes (i.e., Git commits) on the module
MINOR_LINE	The number of unique developers who have contributed less than 5% of the total lines of code on the module
MAJOR_COMMIT	The number of unique developers who have contributed more than 5% of the total code changes (i.e., Git commits) on the module
MAJOR_LINE	The number of unique developers who have contributed more than 5% of the total lines of code on the module
OWN_COMMIT	The proportion of code changes (i.e., Git commits) made by the developer who has the highest contribution of code changes on the module
OWN_LINE	The proportion of lines of code written by the developer who has the highest contribution of lines of code on the module

RQ1の調査方法

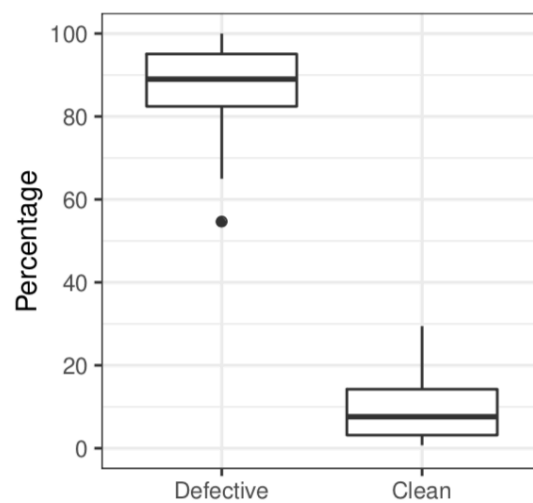


- 1.各プロジェクトのリリースごとのデータセットを用いる
- 2.Defect count datasets（各モジュールに含まれる欠陥数）， Binary defect datasets（各モジュールに欠陥が含まれるか否か）を作成し比較を行なう

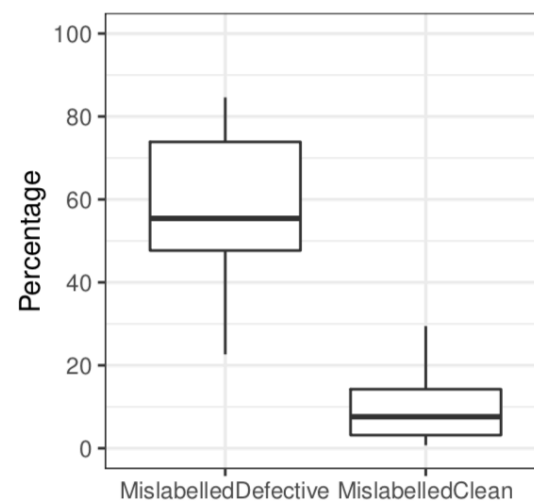
RQ1の結果

Heuristic defects datasetsとRealistic defects datasetsは異なるか？

Heuristic DatasetsがRealistic Datasetsに
一致しなかったモジュールの割合

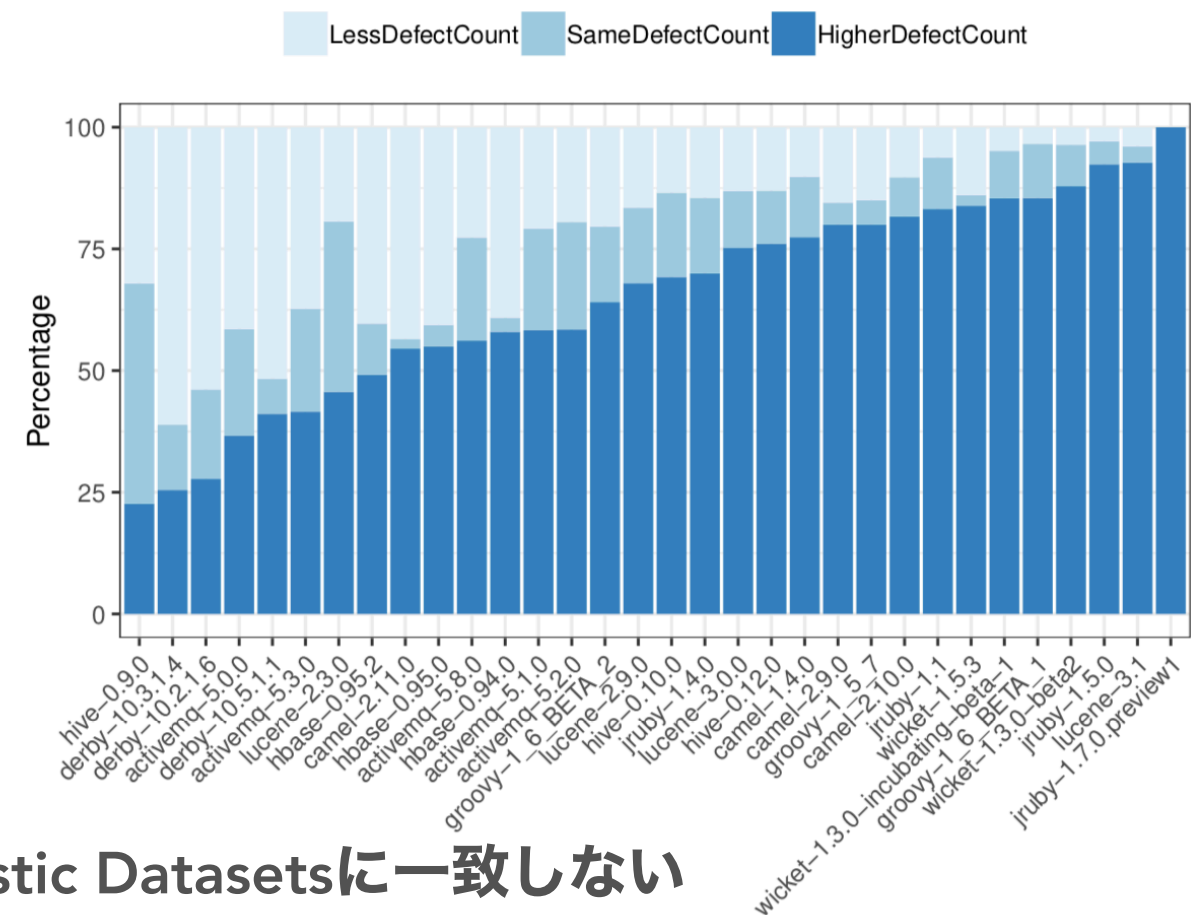


(a) Defect Count Datasets



(b) Binary Defect Datasets

各プロジェクトのモジュールごとの内訳



Heuristic Datasetsは, Realistic Datasetsに一致しない

Defect Count Datasetsは89%一致しない

Binary Defect Datasetsは55% 一致しない

RQ2の調査方法

各Datasetsによって、予測モデルの精度への影響はあるか？

1. Defect Count DatasetsとBinary Defects Datasetsに対して
予測モデルを作成（Bootstrap Samplingを行い試行回数100回）

2. Realistic Approachでテストデータを作成し評価

- 予測モデル

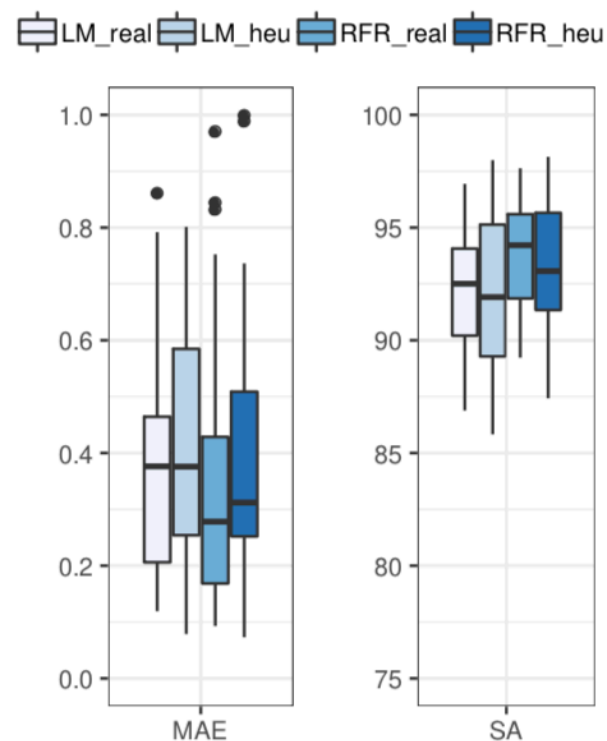
ロジスティック回帰, ランダムフォレスト

- 評価指標

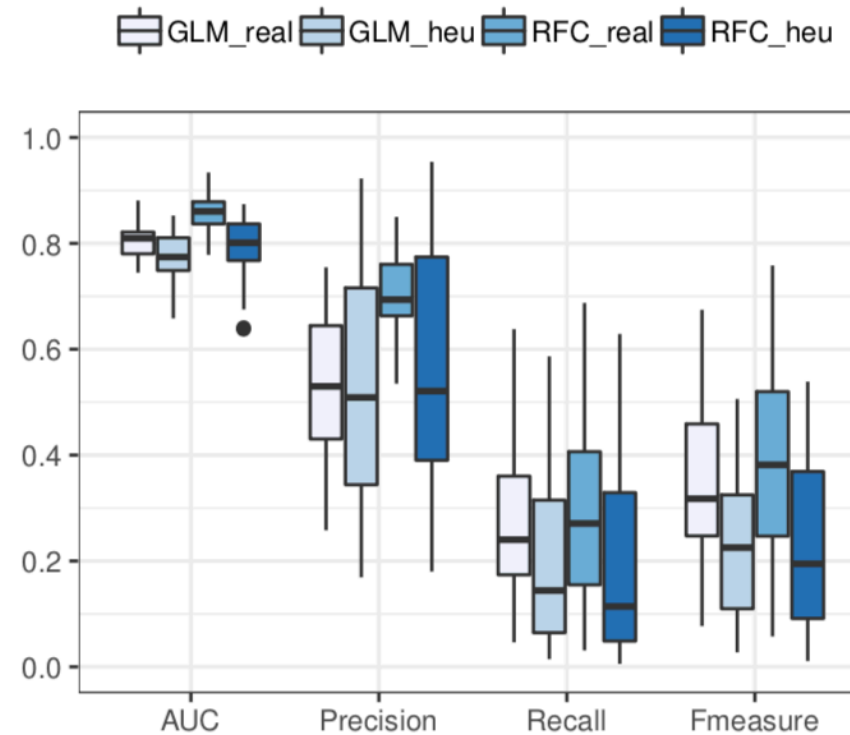
Precision, Recall, F値, AUC, MAE, SA(Standardized Accuracy)

RQ2の結果

各Datasetsによって、予測モデルの精度への影響はあるか？



(a) Defect Count Models



(b) Defect Classification Models

ウィルコクソンの符号付き順位検定の結果、有意差はない

Realistic Approachのほうが、精度が若干高い

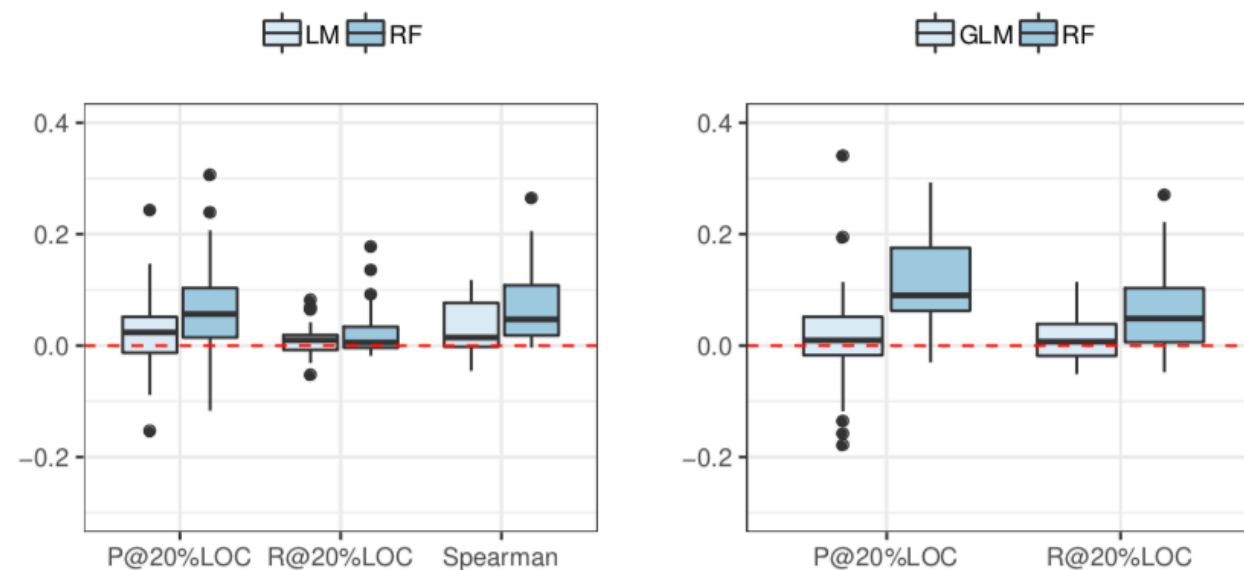
RQ3の調査方法

各Datasetsで予測モデルを作成した場合、欠陥を引き起こすモジュールの
予測ランクへの影響はあるか？

- RQ2とモデルの作成方法は同じ
- 評価指標
 - Precision@20%LOC, Recall@20%LOC
 - ▶ LOC（コード行数）の大きいモジュール上位20%
 - スピアマンの順位相関（予測ランクと実際のランクがどのくらい一致しているか）

RQ3

各Datasetsで予測モデルを作成した場合，欠陥を引き起こすモジュールの
予測ランクへの影響はあるか？



(a) Defect Count Models

(b) Defect Classification Models

Fig. 9: The distributions of defect ranking accuracy difference of (a) defect count models and (b) defect classification models that are constructed using realistic and heuristic defect datasets (i.e., $\text{Measure}_{\text{realistic}} - \text{Measure}_{\text{heuristic}}$).

予測モデルにはDatasetsの違いは，ほとんど影響がない
統計的有意差はなし

まとめと所感

- まとめ

- Heuristic, Realistic Approachで作成させるDatasetsについて調査を行った
 - ▶ Datasetsの性質については大きく異なる
 - ▶ 予測モデルについての影響は少なかった

- 所感

- なぜモデルの予測精度で有意差がでなかったのかの分析, 考察がほしかった