卒 業 研 究 報 告 書

題　目　　ソフトウェア不具合コミット推定手法間の
整合性比較と考察

指導教員　　水野　修　教授

京都工芸繊維大学 工芸科学部 情報工学課程

学生番号　　14122024

氏　　名　　北村　紗也加

平成30年2月14日提出

ソフトウェア不具合コミット推定手法間の整合性比較と考察

平成 30 年 2 月 14 日　　14122024　北村　紗也加

概　要

ソフトウェアの複雑さと重要性は日々増してきており，ソフトウェアの品質を高い水準で保つことが重要視されている．このような現状においてはソフトウェアの品質予測は重要な研究テーマであり，どのような手法で品質予測を行うかに注力されてきた．ソフトウェア不具合コミットを推定する手法では，その評価を行うためには正解データが必要であり，不具合の正解データとして Commit Guru によるソフトウェア不具合コミットの情報が多く利用されている．しかしながら，Commit Guru の不具合コミットの情報の正解データとしての信頼性は不明である．

本研究では，その信頼性に対する検証を行った．ソフトウェア不具合コミット推定手法である SZZ アルゴリズムを用いて，同じ不具合データに対する結果の整合性を比較し，その結果を考察した．Commit Guru，および SZZ アルゴリズムを用いた不具合コミット推定の結果は，不具合コミットを推定したと考えるに十分な可能性を示した．また，Commit Guru と SZZ アルゴリズムの結果の差異において検証を行なった結果，Commit Guru の方がより不具合コミットを推定できると考えられる．これらより，Commit Guru のソフトウェア不具合コミットの情報は不具合の正解データとして可能性を示すものとなった．

目　次

1. 緒言

近年，ソフトウェアの重要性と複雑さは増しつつある．このような現状におかれたソフトウェア開発の現場では，ソフトウェアに混入した不具合を修正するために膨大な時間とエフォートが割かれており，またそれに伴うコストも膨大であるため [1]，ソフトウェアの品質を高い水準で保つことが重要視されている．したがって，最近のソフトウェア工学の研究ではソフトウェアの分析とソフトウェアの品質予測に焦点を当てたものが多く見受けられる [2].

近年，ソフトウェア工学における品質予測の研究を活性化させたものとして，Śliwerski，Zimmermann，Zeller によって提案された，ソースコードやリポジトリデータを用いて不具合を混入したと推定されるコミットを発見する SZZ アルゴリズム [3] がある．SZZ アルゴリズムはバージョン管理システムにおいて変更履歴を辿り，不具合を混入したコミットを特定する．この SZZ アルゴリズムとはまた別のアルゴリズムで不具合コミットを特定するツールを公的に入手可能にしたものとして，Commit Guru[4, 5] 等が存在する．

Commit Guru は，登録された git リポジトリにおける全てのコミットに対し分析を行い，13 のメトリクスを計算し，不具合を混入したコミットを推定する．また，その分析結果を利用者らに無償で提供している．

Commit Guru によるソフトウェア不具合コミット（以降，不具合コミットと呼称）の情報は，昨今の研究において不具合の正解データとして利用されることが多い．しかしながら，Commit Guru が推定する不具合コミットは簡易的な手法を用いており，正解データとしての信頼性は不明である．そこで，本研究では Commit Guru および SZZ アルゴリズムを用いて，分析結果の整合性を比較し，Commit Guru の正解データとしての信頼性について考察を行う．

本紙の構成は次のようになっている．第 2 章では本研究の目的，第 3 章では分析に必要な事前知識，第 4 章では分析対象および事前準備を含む実験手順を述べる．第 5 章での実験結果を受け，第 6 章では考察を行い，第 7 章をもって本紙の結言とする．

2. 研究の目的

2.1 研究の目的

本研究の目的は，Commit Guru と SZZ アルゴリズム，2 つのソフトウェア不具合コミット推定手法の分析結果の整合性を比較し，Commit Guru の正解データとしての信頼性について考察を行うことで，ソフトウェア不具合コミット推定手法の信頼性保証への手がかりを提供することにある．

2.2 研究設問

研究の目的を達成するため，本研究では次に示す研究設問（Research Question）を設け，検証を行う．

- **RQ1: Commit Guru と SZZ アルゴリズムにおいて，ソフトウェア不具合コミット推定の結果は妥当であるか．**
- **RQ2: Commit Guru と SZZ アルゴリズムにおいて，ソフトウェア不具合コミット推定性能の差はどの程度か．**

RQ1 では，ソフトウェア不具合コミット推定手法が推定した不具合コミットが『不具合を混入した』と考えるに妥当であるかを検証する．

RQ2 では，ソフトウェア不具合コミット推定手法が推定する不具合コミットの推定性能にどの程度の差が存在するかを検証する．

3. 用語

3.1 バージョン管理システム

バージョン管理システムとは，コンピュータ上で作成・編集されるファイルの変更履歴を管理するためのシステムである．用語としては，次のようなものが挙げられる．

- リポジトリ
  バージョン管理システムにおいて，ファイルの各バージョンを保持するデータベース．
- コミット
  ローカル環境に存在するファイルに対して行った変更を，リポジトリに存在するファイルに反映させる行為．

本研究では，不具合を混入したであろうコミットを推定する手法を対象に研究を行った．また，バージョン管理システムは，次に述べる Git を使用した．

3.1.1 Git

Git は，Linux カーネルのソースコードの管理に用いる目的で Linus Benedict Torvalds によって開発された，分散型バージョン管理システムである．

3.2 バグトラッキングシステム

バグトラッキングシステムとはプロジェクトのバグを登録し，そのバグの修正状況を追跡するシステムであり，バグの履歴管理や検索を行うことができる．本研究で用いた SZZ アルゴリズムは，バグトラッキングシステムから取得した対象プロジェクトの不具合データを用いて分析を行っており，使用したバグトラッキングシステムは次の 2 つである．

3.2.1 Bugzilla

Bugzilla は，Perl で記述された Mozilla Foundation によって開発されたウェブベースのバグトラッキングシステムである [6]．2018 年 2 月時点において，公開されてい

るだけでも 137 の企業やプロジェクトに利用されている．Bugzilla の不具合データには，不具合 ID，プロジェクト，不具合のステータスなどが含まれており，そのうちから次の情報を使用した．

- Bug ID: 不具合に対して一意に与えられる ID
- Opened: 不具合が発見された日付
- Changed: 不具合が最終的に修正された日付

3.2.2 Apache's JIRA issue tracker

Apache's JIRA issue tracker は Atlassian によって開発された商用の問題トラッキングシステムである（問題とは不具合，機能要求，改善，タスクなど不具合以外にも様々なものを指すが，本研究においてはバグトラッキングシステムとして用いている）[7]．パブリックオープンソースプロジェクトにおいて広く使用されている．Apache's JIRA issue tracker の不具合データには，問題の要約など 50 以上の情報が含まれているが，そのうちから次の情報を使用した．

- Issue key: 問題に対して一意に与えられるキー
- Created: 問題が作成された日付
- Resolved: 問題が解決された日付

3.3 Commit Guru

Commit Guru は Kamei らの変更レベルの不具合予測モデル [8] を Web アプリケーションとして実装したものであり，Rosen らによって公開されている．本研究において，Commit Guru での不具合混入ラベル，メトリクスを分析データの整合性比較に用いる．Commit Guru の機能である (1) 不具合コミットのラベル付け，(2) メトリクスの測定，および (3) ダンプデータの提供について次に述べる．

3.3.1 不具合コミットのラベル付け

Commit Guru はコミットに対し，不具合を混入したコミット (Buggy) とそれ以外のコミット (Clean) を不具合混入ラベルにおいて TRUE/FALSE でラベル付けを行う．不具合コミットは不具合を修正したコミットから推定できる．

始めに，Commit Guru は表 3.1 に示した Hindle らの研究 [9] におけるコミットを分類するためのキーワードリストを元に，コミットメッセージの解析を行い各カテゴリに分類する．ここでは，Corrective に分類されたコミットが不具合を修正しているであろうコミットであるとする．

次に，修正を行ったコミットから不具合を混入したであろうコミットを特定する．まずバージョン管理システムの diff コマンドを用いて，どの行が修正を行ったコミットによって変更されたのかを特定する．ここで変更された行に対して，バージョン管理システムの annotate/blame コマンドを用いることで，それらの修正された行がどのコミットで混入されたのかを特定する．ここで見つかった不具合修正されたソースコードを混入したコミットを，不具合を混入したであろうコミットとしてラベル付けする．

3.3.2 メトリクスの測定

測定されるメトリクスは，Kamei らの不具合予測モデルで利用されているものと同様の数値メトリクス 13 個，および論理メトリクス 1 個から成る変更に関するメトリクスである．

- lines added; la
  変更によって追加されたコードの行数
- lines deleted; ld
  変更によって削除されたコードの行数
- lines total; lt
  変更される前のコードの総行数
- no. subsystem; ns
  変更されたサブシステムの個数
- no. directories; nd
  変更されたディレクトリの個数
- no. files; nf
  変更されたファイルの個数
- no. developers; ndev

表 3.1　コミットの分類におけるカテゴリとその対応キーワード

カテゴリ	キーワード	カテゴリの説明
Corrective	bug, fix, wrong, error, fail, problem, patch	処理の失敗
Feature Addition	new, add, requirement, initial, create	新しい機能の追加
Merge	merge	新しいコミットのマージ
Non Functional	doc	実装していない要件
Perfective	clean, better	パフォーマンスの改善
Preventive	test, junit, coverage, asset	欠陥のテスト

修正されたファイルを変更したことがある人数

- age; age

  修正された全てのファイルにおいて前回の変更から経過した平均日数

- no. unique changes; nuc

  修正されたファイルにおける変更の個数

- experience; exp

  変更を行った開発者の総コミット数

- recent experience; rexp

  age によって重み付けされた experience

- subsystem experience; sexp

  変更されたサブシステムに対するこれまでの変更の数

- entropy; entrophy

  変更が多くのファイルに渡って行われているほど大きくなる値

- fix; fix

  変更が不具合を修正したかどうか (TRUE/FALSE)

3.3.3 ダンプデータの提供

分析した Git リポジトリのダンプデータは CSV 形式で提供されている．このダンプデータには，次の情報が格納されている．ここで，コミットには Author と Committer が存在し，Author はオリジナルのコードを書いた人，Commtter はコミットした人のことを指す.

- commit_hash: コミットハッシュ
- author_name: Author の名前
- author_date_unix_timestamp: コミットの UNIX タイムスタンプ
- author_email: Author の E メールアドレス
- author_date: コードを書いた日付
- commit_message: コミットメッセージ
- classification: 分類カテゴリ
- linked: 不具合を混入したコミットとリンクできたか (TRUE/FALSE)

- contains_bug: 不具合混入ラベル (TRUE/FALSE)
- fixes: そのコミットの不具合を修正したコミットのリスト
- fileschanged: そのコミットにおいて変更されたファイル
- 3.3.2 節において述べた 14 個の計測したメトリクス
- glm_probability: 今後，修正が必要になる変更を行った可能性
- repository_id: コミットを行ったリポジトリの ID

本研究では，このダンプデータを用いて分析を行っている．

3.4 SZZ アルゴリズム

SZZ アルゴリズムはソフトウェア不具合コミット推定手法の中で最もよく知られている手法であり，提案者の Śliwerski，Zimmermann，Zeller の頭文字をとってこう呼ばれる．

本研究で使用した SZZ アルゴリズムは，バージョン管理システムのバージョンアーカイブとバグトラッキングシステムの不具合データを用いて，次の 3.4.1 節，3.4.2 節で述べる 2 つのステップを経ることによりソフトウェア不具合コミットの推定を行う．

3.4.1 修正された箇所の特定

バージョン管理システムのバージョンアーカイブには，変更に関する情報が含まれている．バージョン間において同じ意図で行われているの変更を特定するため，バージョンアーカイブから正規表現を用いて，不具合が混入されたコミットの日時，不具合を最後に修正されたコミットの日時を抽出し，バグトラッキングシステムの不具合データと照らし合わせる．抽出されたデータが不具合の修正と予測されるのであれば，この不具合を修正したであろうコミットに対し，修正したことを示す FIX タグをつける．

バージョン管理システムのバージョンアーカイブには変更の目的が欠けているため，行われた修正が不具合を修正したのか，機能追加であるかの判断することができない．ログメッセージのみから目的を予測することは出来る [10] が，不具合デー

タには簡単な説明が載っている不具合レポート，不具合のステータスや解決策が格納されているため，不具合データを組み合わせることで予測の精度向上を図る．

3.4.2 不具合が混入された箇所の特定

FIX タグがつけられたコミットを取得する．このコミットにおいて変更されたファイルに対し，バージョン管理システムの diff コマンドを用いてコミットにおける変更箇所を抽出する．

もし変更箇所が見つからなければ FIX タグの削除を行い，変更箇所が見つかれば，変更箇所はどのコミットで混入されたのかを特定する．変更箇所に対してバージョン管理システムの blame コマンドと正規表現を用いて，その変更箇所が混入したコミットの日時を抽出する．抽出されたデータからコミットが不具合を混入したと予測された場合には，不具合コミットであることを示す BUG タグをつける．

3.4.3 データフォーマット

本研究において使用した SZZ アルゴリズムに適用するバグトラッキングシステムの不具合データは，CSV 形式において次に述べる 3 つのカラムを必要とし，それぞれが指定されたデータフォーマットに則る必要がある．

- bug issue–id: 不具合 ID
  不具合 ID は数字のみから成り立つ．
- open date: 不具合のステータスが OPEN になった日時
  YYYY–MM–DD hh:mm:ss となっていなければならない．
- closed or fixed date: 不具合のステータスが CLOSED または FIXED になった日時
  YYYY–MM–DD hh:mm:ss となっていなければならない．

3.5 検定

3.5.1 ウィルコクソンの順位和検定

ウィルコクソンの順位和検定 (Wilcoxon rank sum test) とは，ノンパラメトリック検定のひとつであり,『対応のない 2 群が同じ分布の母集団から構成されている』と

する帰無仮説に基づいて検定する．

データサイズ n_1 のデータ D_1，およびデータサイズ n_2 のデータ $N_2(n_1 \leq n_2)$ から成る 2 つのサンプルデータ群に対し，両郡のデータの値が小さい順に順位を割り当て，D_1, D_2 のそれぞれの順位和 R_1, R_2 を求める．同じ値のデータが存在した場合には，それらが異なると考えた場合の順位の平均値を割り当てる．このとき，D_1, D_2 の順位和の合計は次のようになる．

$$R_1 + R_2 = \frac{N(N+1)}{2} \tag{3.1}$$

これより，両郡のデータサイズ，順位和を用いて，次の式を用いて統計量を求める．

$$U_1 = R_1 - \frac{n_1(n_1+1)}{2} \tag{3.2}$$

$$U_2 = R_2 - \frac{n_2(n_2+1)}{2} \tag{3.3}$$

3.2 式，3.3 式によって求めた U_1, U_2 において小さい方の値を U とする．U は有意性を調べるときに用いられる．サンプルサイズが大きい場合，U は正規分布に近似できる．このとき，

$$z = \frac{U - m_U}{\sigma_U} \tag{3.4}$$

ここで，m_U および σ_U は U の平均および標準偏差であり，有意性を調べることができる標準正規偏差である．m_U および σ_U は

$$m_U = \frac{n_1 n_2}{2} \tag{3.5}$$

$$\sigma_U = \sqrt{\frac{n_1 n_2 (n_1 + n_2 + 1)}{12}} \tag{3.6}$$

タイ値が存在した場合，次の式で σ を補正する必要がある．

$$\sigma_{\mathrm{corr}} = \sqrt{\frac{n_1 n_2}{12}\left((n+1) - \sum_{i=1}^{k} \frac{{t_i}^3 - t_i}{n(n-1)}\right)} \tag{3.7}$$

ここで，$n = n_1 + n_2$ であり，t_i はランク i を共有するタイ値の数であり，k はランクの数である。

本研究において使用した Commit Guru は，Buggy/Clean なコミットのグループ化を行い，Python の rpy2 パッケージを用いて，ウィルコクソンの順位和検定における

測定したメトリクスの有意差を検定している．この結果はCommit Guruから提供されているダンプデータには格納されていないが，Webページから確認することが出来る．

3.5.2 マン・ホイットニーのU検定

マン・ホイットニーのU検定 (MannWhitney U test) とは，ノンパラメトリック検定のひとつであり,『対応のない2群が同じ分布の母集団から構成されている』とする帰無仮説に基づいて検定する．実質，ウィルコクソンの順位和検定と同じ方法であり，この2つをまとめて，マン・ホイットニー・ウィルコクソン検定とも呼ばれる．

本研究においては，SZZアルゴリズム，Commit Guruの結果の整合性比較のため，Pythonのscipyパッケージを用いてこのマン・ホイットニーのU検定を使用した．

3.6 留意点

留意点として，Commit Guruでは，コミットに対する不具合混入ラベルとして「真値」および「推定値」を持っている．ここで，推定値はCommit Guru独自の推定アルゴリズムによる値となっている．これに対し，SZZアルゴリズムはコミットに対する不具合混入ラベルとして「真値」のみ持っている．また，これらの真値は確実な正解データではなく，あくまでも「その可能性が高い」ということを示唆している．

本研究においては，Commit Guru，SZZアルゴリズムの両手法において，不具合混入ラベルとして「真値」を採用している．

4. 実験

4.1 対象プロジェクト

本実験では，次に述べるプロジェクトを対象に分析を行った．

4.1.1 Apache Log4j

Apache Log4j (以降 Log4j と呼称) は，Java のロギングツールである．元は Ceki Gülcü によって開発されていたが，現在は Apache ソフトウェア財団のプロジェクトの 1 つとなっている．バグトラッキングシステムには Bugzilla を用いていたが，現在は Apache's JIRA issue tracker に移行しており，移行に際して登録 Git リポジトリも新しいものに変わっている．

Commit Guru では，Bugzilla に登録されている Git リポジトリを対象に分析されているため，本研究に用いた SZZ でも Bugzilla に登録されている Git リポジトリ，および Bugzilla から得られた不具合データを用いて Log4j の分析を行っている．

4.1.2 Apache OpenJPA

Apache OpenJPA (以降 OpenJPA と呼称) は，Java Persistence API 仕様のオープンソース実装の 1 つであり，オブジェクト関係マッピングの 1 つでもある．元は SolarMetric 社によって開発されたが，SolarMetric 社を買い取った BEA システムズによって Apache Software Foundation に寄贈され，現在は Apache ソフトウェア財団のトッププロジェクトの 1 つとなっている．

4.1.3 Apache Camel

Apache Camel (以降 Camel と呼称) は、エンタープライズ統合パターンに基づいた Java ベースのオープンソース統合フレームワークであり．Apache ソフトウェア財団のトッププロジェクトの 1 つとなっている．

4.1.4 Apache HBase

Apache HBase (以降 HBase と呼称) は，列指向データベース管理システムである．元は Powerset 社におけるプロジェクトとして開始されたが，Apache ソフトウェア財団の Hadoop プロジェクトの一部として開発されるようになり，現在では HBase そのものが Apache ソフトウェア財団のトッププロジェクトの 1 つとなっている．

4.2 事前準備

4.2.1 Git リポジトリ

分析対象となるプロジェクトの Git リポジトリをダウンロードする．

本研究では，Log4j，OpenJPA，Camel，HBase の 4 つのプロジェクトにおいて分析を行なっているため，それぞれ次の表 4.1 に示すリポジトリのダウンロードを行った．

4.2.2 バグトラッキングシステムの不具合データ

Log4j は Bugzilla から，OpenJPA，Camel，HBase は Apache's JIRA issue tracker からそれぞれ CSV 形式の不具合データを取得している．

それぞれの不具合データ取得の際の条件を表 4.2，表 4.3 に示す．

どちらのバグトラッキングシステムから得た不具合データにおいても，3.4.3 節で述べたデータフォーマットに則り，3.2.1 節，3.2.2 節において述べた情報のみになるようカラムの削除，および必要に応じてカラムデータの書き換えを行っている．

また，各プロジェクトにおけるバグトラッキングシステムから取得した不具合データとは表 4.4 のようになっている．

4.2.3 Commit Guru のダンプデータ

Commit Guru によって提供されている対象プロジェクト，Log4j，OpenJPA，Camel，HBase のダンプデータをダウンロードする．得られたダンプデータの詳細を表 4.5 に示す．

表 4.1　対象プロジェクトとその Git リポジトリ

Project	Repository URL
Log4j	https://github.com/apache/log4j
OpenJPA	https://github.com/apache/openjpa.git
Camel	https://github.com/apache/camel
HBase	https://github.com/apache/hbase.git

表 4.2　Bugzilla から取得した不具合データの取得条件

Projects	resolved	Resolution	Component	status
Log4j	- 2018/01/22	FIXED	Appender, chainsaw, Companions, Configurator, Layout	RESOLVED, VERIFIED, CLOSED

表 4.3　Apache's JIRA issue tracker から取得した不具合データの取得条件

project	resolved	Issue type	status
OpenJPA	- 2018/01/17	bug	Closed
Camel	- 2018/01/22	bug	Closed
HBase	- 2018/01/29	bug	Closed

表 4.4　バグトラッキングシステムから取得した不具合データ

project	総ファイル数	総不具合データ数	データ取得日
Log4j	1	305	2018/01/22
OpenJPA	2	1162	2018/01/17
Camel	2	1457	2018/01/22
HBase	7	5466	2018/01/29

表 4.5　得られた Commit Guru のダンプデータ

Project	総データ数	Buggy rate	Date dumped using data
Log4j	3275	45.6%	2018/01/17
OpenJPA	4864	11.1%	2018/01/10
Camel	30739	20.7%	2018/01/10
HBase	14745	31.9%	2018/01/23

4.3 実験手順

以降において，指定したコミットおよびコミットのペアを次のように呼称する．

- SZZ，Commit Guru によって Buggy/Clean であると推定されたコミットをそれぞれ BUG_{szz}，BUG_{guru}，$CLEAN_{szz}$，$CLEAN_{guru}$ と呼称する．
- (BUG_{szz}, $CLEAN_{szz}$), (BUG_{guru}, $CLEAN_{guru}$) のペアをそれぞれ CF_{szz}, CF_{guru} と呼称する．
- Commit Guru，SZZ アルゴリズムにおいて，推定結果が異なるコミットをそれぞれ $DIFF_{guru}$, $DIFF_{szz}$ と呼称する．ここで，下付き文字は Buggy であると推測した手法である．すなわち，$DIFF_{guru}$ は，Commit Guru において Buggy と推定されたが SZZ アルゴリズムでは Clean と推定されたコミット，$DIFF_{szz}$ は SZZ アルゴリズムによって Buggy と推定されたが Commit Guru では Clean と推定されたコミットである．

実験手順を次に述べる．

1. SZZ アルゴリズムを，不具合データの CSV ファイルが入っているディレクトリのパスと Git リポジトリのディレクトリのパスを与え実行する．SZZ アルゴリズムによって，コミットに対し BUG タグのタグ付けが行われる．
2. 処理が終了した後，対象プロジェクトの git リポジトリ内で grep コマンドを用いて，コミットに対して付けられた不具合を示す BUG タグを出力する．
3. 出力された全てのタグに対しバージョン管理システムの show コマンドを用いて，そのタグが付けられたコミットの情報を出力する．
4. 出力されたコミットから，コミットハッシュ，コミットに対し付けられた BUG タグ，Tagger，Tagger の E メールアドレスを正規表現を用いて抽出する．Tagger, Tagger の E メールアドレスが著者と一致するデータから，BUG_{szz} を作成する．
5. 抽出したデータのコミットハッシュと一致するコミットを Commit Guru のダンプデータから抽出し，BUG_{szz} を作成する．一致しなかったコミットも抽出し，$CLEAN_{szz}$ を作成する．同様に，$DIFF_{szz}$，$DIFF_{guru}$ も作成する．
6. Commit Guru のダンプデータから BUG_{guru}，$CLEAN_{guru}$ を作成する．
7. BUG_{szz}，$CLEAN_{szz}$，BUG_{guru}，$CLEAN_{guru}$, $DIFF_{szz}$，$DIFF_{guru}$，の中央値を測定

する．

8. CF_{szz}，CF_{guru} においてマン・ホイットニーの U 検定を行う．$DIFF_{szz}$，$DIFF_{guru}$ 間においてもマン・ホイットニーの U 検定を行う．

5. 実験結果

5.1 SZZ アルゴリズム

SZZ アルゴリズムの実行結果を表 5.1 に示す．SZZ アルゴリズムは 1 つのコミットに対し複数の BUG タグを付けるため，Buggy rate として BUG タグが付けられたコミット数と，コミットについた BUG タグ数で重み付けしたコミット数の 2 つを算出した．

5.2 各メトリクスの中央値と有意差

表において示された p 値より得られた有意差を，p 値の右側に*を用いて表記した．*，**，***はそれぞれ $p < 0.05$，$p < 0.01$，$p < 0.005$ を示す．

5.2.1 ソフトウェア不具合コミットの中央値

Log4j，OpenJPA，Camel，HBase の中央値をそれぞれ表 5.2，表 5.3，表 5.4，表 5.5 に示す．ここで，fix rate は次の式に則り計算を行なった．

$$\text{fix rate} = \frac{(\text{fix が TURE である数})}{\text{全データ数}} \tag{5.1}$$

これらより，各プロジェクトの中央値において次のような傾向が存在した．

- ns，nd，ndev

  各プロジェクト通じ，Buggy/Clean と推定されたコミット間において特に大きな差は見られなかった．しかしながら，Buggy と推定されたコミットが Clean と推定されたコミットの値を下回ることはなかった．

- nf

  Buggy と推定されたコミットは，Clean と推定されたコミットに対しておおよそ 2〜3 倍の値をもつ．

- entrophy

  Buggy と推定されたコミットは entrophy が高く，最低でもおおよそ 1 以上の値である．これに対し，Clean と推定されたコミットは最高でも 0.7 未満に収まっている．

表 5.1　SZZ アルゴリズムから得られた結果

project	総 BUG タグ数	総コミット数	Buggy rate	Weighting Buggy rate
Log4j	531	296	9.0%	16.2%
OpenJPA	3156	1301	26.8%	64.9%
Camel	4473	2171	7.1%	14.6%
HBase	4257	2408	16.5%	28.9%

表 5.2　Log4j における各メトリクスの中央値

metric	BUG_{szz}	BUG_{guru}	$CLEAN_{szz}$	$CLEAN_{guru}$
ns	1.00000	1.00000	1.00000	1.00000
nd	3.00000	2.00000	1.00000	1.00000
nf	6.00000	2.00000	1.00000	1.00000
entrophy	1.79663	0.96496	0.00000	0.00000
la	116.50000	49.00000	15.00000	8.00000
ld	28.00000	13.00000	5.00000	3.00000
lt	199.35000	240.00000	256.00000	258.25000
ndev	5.00000	7.00000	7.00000	7.00000
age	17.31489	12.21238	5.94594	3.84601
nuc	45.50000	30.00000	22.00000	17.00000
exp	2829.50000	1429.00000	1279.50000	1311.25000
rexp	1.03343	1.06509	1.11733	1.15206
sexp	823.00000	653.00000	506.00000	420.50000
fix rate	30.41%	34.45%	31.25%	28.43%

表 5.3　OpenJPA における各メトリクスの中央値

metric	BUG_{szz}	BUG_{guru}	$CLEAN_{szz}$	$CLEAN_{guru}$
ns	2.00000	2.00000	1.00000	1.00000
nd	2.00000	3.00000	1.00000	1.00000
nf	4.00000	5.00000	1.00000	1.00000
entrophy	1.29974	1.71893	0.00000	0.00000
la	82.00000	139.00000	9.00000	12.00000
ld	8.00000	15.00000	3.00000	3.00000
lt	459.50000	443.50000	377.00000	388.00000
ndev	12.00000	12.00000	17.00000	17.00000
age	22.53412	17.36152	14.89322	16.77649
nuc	42.00000	48.00000	22.00000	25.00000
exp	817.00000	620.50000	896.00000	909.75000
rexp	1.02119	1.01864	1.04396	1.03614
sexp	88.00000	88.00000	99.00000	97.00000
fix rate	11.84%	14.63%	15.24%	14.29%

表 5.4　Camel における各メトリクスの中央値

metric	BUG_{szz}	BUG_{guru}	$CLEAN_{szz}$	$CLEAN_{guru}$
ns	1.00000	1.00000	1.00000	1.00000
nd	3.00000	3.00000	2.00000	1.00000
nf	6.00000	5.00000	2.00000	2.00000
entrophy	1.97566	1.66573	0.68404	0.32985
la	120.00000	105.00000	13.00000	9.00000
ld	15.00000	10.00000	3.00000	2.00000
lt	127.33333	121.00000	136.63889	141.00000
ndev	12.00000	18.00000	32.00000	32.00000
age	13.71051	15.45778	5.76493	4.68852
nuc	51.00000	34.00000	24.00000	24.00000
exp	4988.50000	5651.25000	5621.00000	5481.00000
rexp	1.03170	1.02234	1.06519	1.08393
sexp	1370.00000	1315.50000	764.00000	691.00000
fix rate	12.76%	16.84%	16.38%	15.94%

表 5.5　HBase における各メトリクスの中央値

metric	BUG_{szz}	BUG_{guru}	$CLEAN_{szz}$	$CLEAN_{guru}$
ns	2.00000	2.00000	1.00000	1.00000
nd	3.00000	3.00000	1.00000	1.00000
nf	4.00000	4.00000	2.00000	1.00000
entrophy	1.46813	1.42000	0.43275	0.00000
la	112.00000	116.50000	13.00000	9.00000
ld	29.00000	27.00000	4.00000	3.00000
lt	811.81538	773.93333	734.66667	739.00000
ndev	14.00000	28.00000	35.00000	32.00000
age	11.56159	17.73020	13.32322	10.40514
nuc	306.50000	178.00000	51.00000	45.00000
exp	1306.25000	967.00000	1026.50000	1160.50000
rexp	1.20463	1.06213	1.07579	1.11162
sexp	638.50000	411.00000	257.00000	259.00000
fix rate	15.49%	18.69%	19.79%	19.27%

- la, ld

  各プロジェクトを通じ，Buggy と推定されたコミットでは la は約 8〜12 倍，ld は約 3〜9 倍の値が Clean と推定されたコミットに対して得られた．

- lt

  lt はプロジェクト間において値の差が大きく，全プロジェクトに通じる基準値となるものが設定し難い．

- age

  基本的に Buggy と推定されたコミットは Clean と推定されたコミットよりも age が高く，最低でも 10 以上の値である．

- nuc

  Buggy/Clean と推定されるコミット間において大きな差が見られた．プロジェクト間において値の差が大きく，全プロジェクトに通じる基準値となるものは設定し難いが，Buggy と推定されたコミットにおいて，Clean と推定されたコミットに対しておおよそ 2〜5 倍の値が得られている．

- exp

  プロジェクト間によって値にばらつきが見られた．CF_{guru} 間の比較に対し，OpenJPA, HBase においては Buggy と推定されたコミットの方が値が小さく，CF_{szz} 間の比較に対しては，OpenJPA, Camel において Buggy と推定されたコミットの方が値が小さい結果が得られた．

- rexp

  全てにおいて，1〜2 の間に値が収まっている．HBase における CF_{szz} 以外のプロジェクトおよび比較においては，全て Buggy と推定されるコミットの方が値が小さい．

- sexp

  プロジェクト間によって値にばらつきが見られた．また，OpenJPA 以外は Buggy と推定されたコミットの方が値が大きくなっている．

- fix

  プロジェクト間によって確率にばらつきが見られた．CF_{guru} においては HBase 以外のプロジェクトで Buggy と推定されたコミットの方が fix rate が高いことに対し，CF_{szz} においては全てのプロジェクトで Buggy と推定されるコミット

の方が低い結果となった．

5.2.2 ソフトウェア不具合コミットの有意差

Log4j，OpenJPA，Camel，HBase のマンホイットニーの U 検定を用いて CF_{guru}，CF_{szz} 間において比較を行った．その p 値および有意差をそれぞれ表 5.6，表 5.7，表 5.8，表 5.9 に示す．また，fix は値ではなく真値のため，fix rate における有意差は評価していない．

また，対象プロジェクトを通じて，CF_{guru}，CF_{szz} 間において有意差が存在しなかったペアのメトリクスは次である．

- Log4j
  - CF_{guru}: exp
- Openjpa
  - CF_{guru}: lt, age
  - CF_{szz}: exp, sexp
- Camel
  - CF_{szz}: sexp

表 5.6　Log4j における各メトリクスの p 値および有意差

metric	CF_{guru}	CF_{szz}
ns	1.19047e-32***	4.09322e-34***
nd	8.85456e-76***	8.15276e-53***
nf	2.14142e-72***	2.79911e-54***
entrophy	3.86561e-71***	3.92893e-50***
la	5.90907e-100***	5.05739e-50***
ld	1.09431e-42***	1.97774e-19***
lt	4.57751e-03***	2.21123e-07***
ndev	2.71829e-02*	3.86223e-08***
age	2.72800e-11***	9.27818e-05***
nuc	1.63918e-13***	2.56570e-11***
exp	1.63944e-01	4.19908e-05***
rexp	1.08433e-05***	1.94072e-06***
sexp	5.07598e-24***	5.19319e-05***

表 5.7　OpenJPA における各メトリクスの p 値および有意差

metric	CF_{guru}	CF_{szz}
ns	1.50020e-65***	9.38346e-93***
nd	2.74583e-77***	7.03391e-105***
nf	9.82733e-78***	2.54750e-106***
entrophy	1.70670e-69***	6.53376e-95***
la	2.81904e-99***	1.03822e-158***
ld	2.61118e-43***	2.49774e-47***
lt	9.09889e-02	9.72099e-05***
ndev	1.30431e-24***	4.46056e-61***
age	2.98843e-01	9.28504e-03**
nuc	1.03982e-17***	2.11158e-27***
exp	1.67934e-05***	4.84732e-01
rexp	1.11599e-07***	5.29152e-13***
sexp	5.99722e-02	2.14531e-01

表 5.8　Camel における各メトリクスの p 値および有意差

metric	CF_{guru}	CF_{szz}
ns	7.60597e-89***	1.14164e-108***
nd	0.00000e+00***	0.00000e+00***
nf	0.00000e+00***	0.00000e+00***
entrophy	0.00000e+00***	0.00000e+00***
la	0.00000e+00***	0.00000e+00***
ld	1.70037e-279***	1.31387e-154***
lt	2.30007e-42***	2.70685e-07***
ndev	1.50950e-213***	0.00000e+00***
age	6.01582e-64***	1.85283e-13***
nuc	3.92469e-09***	1.05179e-33***
exp	2.14981e-06***	5.16843e-15***
rexp	9.71377e-131***	1.51230e-24***
sexp	7.08588e-14***	4.24036e-01

表 5.9　HBase における各メトリクスの p 値および有意差

metric	CF_{guru}	CF_{szz}
ns	1.32752e-235***	4.70019e-168***
nd	0.00000e+00***	1.35104e-297***
nf	0.00000e+00***	6.95661e-308***
entrophy	0.00000e+00***	3.48292e-258***
la	0.00000e+00***	0.00000e+00***
ld	0.00000e+00***	1.12541e-178***
lt	4.28478e-03***	3.93796e-04***
ndev	2.20818e-22***	0.00000e+00***
age	2.79573e-27***	3.98334e-05***
nuc	3.20946e-146***	1.33739e-154***
exp	1.31244e-02*	2.75295e-07***
rexp	2.39887e-25***	3.84358e-33***
sexp	4.11822e-21***	2.02175e-66***

6. 考察

6.1 RQ1: Commit Guru と SZZ アルゴリズムにおいて，ソフトウェア不具合コミット推定の結果は妥当であるか.

ソフトウェア不具合コミットは，何らかの形で不具合を混入していないコミットとの間に差が生じていると考えられる．ゆえに，ソフトウェア不具合コミット推定手法においても，その推定されたソフトウェア不具合コミットとそれ以外のコミットの結果には差が生じていると思われる.

Commit Guru，および SZZ アルゴリズムによる不具合コミットの推定結果の優位性および整合性についてそれぞれ検証を行った結果を踏まえ，Commit Guru，および SZZ アルゴリズムにおいて不具合コミットと推定されたコミットが本当に不具合を混入していると考えるに妥当であるかについて考察を行う.

RQ1 を回答するに際し，次の点に関し検証を行った.

- BUG_{szz}，BUG_{guru}，$CLEAN_{szz}$，$CLEAN_{guru}$ の測定された各メトリクスの中央値から，不具合コミットと推定されるコミットにおいて得られた傾向は，不具合コミットを検出したと仮定した場合において，妥当であると考えられるか.
- BUG_{szz} と $CLEAN_{szz}$，BUG_{guru} と $CLEAN_{guru}$ をマン・ホイットニーの U 検定を用いて比較したとき，測定された各メトリクスに有意差は存在するか.

5.2.1 節において得られた対象プロジェクトを通じて各メトリクスにおける中央値の傾向について，それぞれ考察を行う.

- ns, nd, ndev

  1 コミットにおいてディレクトリ，サブシステムの数がより修正されている，すなわち，ns, nd がより大きいコミットは，Buggy である可能性が高いと考えられる [11]．また，コーディングデザインの違いの観点から，ファイルの開発に携わった人数，すなわち ndev が大きいほど Buggy である可能性が高いと考えられる [12].

  対象プロジェクトの中央値に大きな差は見られなかったが，下回ることはなかったことから，ns, nd, ndev の観点において BUG_{guru}, BUG_{szz} が不具合コミッ

トである可能性は存在する．

- nf

  多くのファイルを修正している，すなわち nf が大きいコミットは，Buggy である可能性が高いと考えられる [13].

  両手法において Buggy と推定されたコミットは，Clean と推定されたコミットに対し 2〜3 倍の値を持つことから，nf の観点において BUG_{guru}, BUG_{szz} が不具合コミットである可能性は存在する．

- entrophy

  コミットの entrophy が大きいことは開発者が多くのファイルに分散された変更箇所を探し出して変更した，ということを指す．これより，entrophy が大きいほど Buggy である可能性が高いと考えられる [14, 15].

  両手法において，Buggy と推定されたコミットは entrophy が高く，最低でもおおよそ 1 以上の値であり，Clean と推定されたコミットは最高でも 0.7 未満に収まっていることから，entrophy の観点において BUG_{guru}, BUG_{szz} が不具合コミットである可能性は存在する．

- la, ld

  1 コミットにおいてより多くの行が追加，削除される，すなわち la, ld が大きいコミットは，Buggy である可能性が高いと考えられる [16, 17].

  各プロジェクトを通じ，両手法において Buggy と推定されたコミットでは la は約 8〜12 倍，ld は約 3〜9 倍の値が Clean と推定されたコミットに対して得られたことから，la, ld の観点において BUG_{guru}, BUG_{szz} が不具合コミットである可能性は存在する．

- lt

  コード行が多いファイルは，そのファイル内においてより多くの不具合を混入したコミットがなされていると考えられる．すなわち，lt が大きいコミットは，Buggy である可能性が高いと考えられる [18].

  両手法を通じ，Log4j，Camel において，Clean と推定されたコミットの方が値を上回った．また，Buggy と推定されたコミットの方が値を上回った OpenJPA においては有意差も見られなかった．

- age

前に行われた変更よりも，最近の変更の方が不具合を混入していると考えられる [19]．ゆえに，ファイルが頻繁に更新されているコミット，すなわち age が小さいコミットは，Buggy である可能性が高いと考えられる．

対象プロジェクトの全てに対し，Clean と推定されたコミットの方が小さい結果となった．これより，age の観点において，BUG_{guru}, BUG_{szz} は不具合コミットである可能性は低いと考えられる．

- nuc

  nuc が大きいことは，開発者が以前における変更を探し出さなければならないことを指し，それゆえ不具合を混入する可能性が高いことが考えられる [15]．

  対象の全プロジェクトに対し，Buggy と推定されたコミットは Clean と推定されたコミットに対しておおよそ 2〜5 倍の値が得られている．これより，nuc の観点において，BUG_{guru}，BUG_{szz} は Buggy である可能性が存在すると考えられる．

- exp

  exp が大きいことは，開発者の経験が豊富であることを指すため，Buggy である可能性は低いと考えられる [11]．

  プロジェクト間によって値が異なるものの，BUG_{guru} は OpenJPA，HBase において $CLEAN_{guru}$ よりも値が小さく，BUG_{szz} は Log4j，HBase において $CLEAN_{szz}$ よりも値が大きい．CLEAN と推定されるコミットの方が値が大きいプロジェクトにおいても，その値の差は Commit Guru の方が小さく，OpenJPA においては有意差が存在しなかった．これより，exp の観点において，BUG_{guru}，BUG_{szz} において Buggy である可能性は考えられ難い．

- rexp

  rexp が大きいことは，最近そのファイルに対しよく変更を行なっている人物が開発を行なっていることを指す．そのような人物は他と比べてそのファイルに対しての知識が深いと考えられるため，rexp が大きいコミットは Buggy である可能性が低いと考えられる [11]．

  HBase における CF_{szz} 以外のプロジェクトおよび両手法の CF_{szz}, CF_{guru} において，Buggy と推定されたコミットの方が値が小さい結果が得られた．これより，rexp の観点において，BUG_{guru} と BUG_{szz} が Buggy である可能性は存在すると

考えられる．

- sexp

  sexp が大きいことは，サブシステムについてよく知っている人物が変更を行なっていることを指すため，sexp が大きいコミットは Buggy である可能性が低いと考えられる [11]．プロジェクト間によって値が異なるものの，OpenJPA 以外のプロジェクトにおいて Buggy の方が値が大きい結果が得られた．これより，sexp の観点において，BUG_{guru} と BUG_{szz} が Buggy である可能性は低いと考えられる．しかしながら，OpenJPA, Camel においては Commit Guru で有意差が存在しないため，BUG_{guru} については一概には言えない．

- fix

  新機能の実装による変更よりも，欠陥の修正による変更の方が不具合を混入しやすいことが知られている．fix が TURE であるということは，初期実装において欠陥が存在していたことを指すため，fix の確率が高いということは，Buggy である可能性が高いと考えられる．BUG_{guru} は OpenJPA，HBase 以外のプロジェクトにおいては $CLEAN_{guru}$ よりも高い確率を持っている一方，BUG_{szz} はどのプロジェクトにおいても $CLEAN_{szz}$ より確率が低い結果となった．これより，fix の観点において，BUG_{guru} が Buggy である可能性は存在すると考えられるが，BUG_{szz} においてはその可能性は低いと考えられる．

また，5.2.2 節より，対象となっている全プロジェクトの大半のメトリクスにおいて有意差が得られている．有意差が存在しないメトリクスの数は非常に少なく，これが推定結果の妥当性に影響を与える可能性は考えられにくい．

これを踏まえ，RQ1 について回答を行う．

Commit Guru，および SZZ アルゴリズムのどちらにおいても，Buggy/Clean と推定されたコミットから得られた中央値には Buggy と考えるに十分な可能性が存在する傾向が散見される．また，大半のメトリクスには有意差が存在することから，Commit Guru は正解データとして使用できる可能性は十分に存在すると考えられる．

6.2 RQ2: Commit Guru と SZZ アルゴリズムにおいて，ソフトウェア不具合コミット推定性能の差はどの程度か．

Commit Guru，SZZ アルゴリズムにおける Buggy と推定されたコミット群のコミットは次のいずれかに分類される．

- Commit Guru,SZZ アルゴリズムの両方において Buggy であると推定されたコミット
- Commit Guru では Buggy と推定されたが SZZ アルゴリズムでは Clean と推定されたコミット $\mathrm{DIFF}_{\mathrm{guru}}$
- SZZ アルゴリズムでは Buggy と推定されたが Commit Guru では Clean であると推定された $\mathrm{DIFF}_{\mathrm{szz}}$

ソフトウェア推定性能の差をより顕著に示すため，ここでは $\mathrm{DIFF}_{\mathrm{guru}}$ および $\mathrm{DIFF}_{\mathrm{szz}}$ の比較を行なった．

また，RQ2 を回答するに際し，次の点において検証を行った．

- $\mathrm{DIFF}_{\mathrm{guru}}$, $\mathrm{DIFF}_{\mathrm{szz}}$ の中央値からどのような傾向が得られるか．
- $\mathrm{DIFF}_{\mathrm{guru}}$, $\mathrm{DIFF}_{\mathrm{szz}}$ の値の分布はどのようになっているか．

$\mathrm{DIFF}_{\mathrm{szz}}$, $\mathrm{DIFF}_{\mathrm{guru}}$ の中央値および有意差を表 6.1, 表 6.2, 表 6.3, 表 6.4 に示す．ここで，表において示された p 値より得られた有意差を，p 値の右側に*を用いて表記した．*，**，***はそれぞれ $p < 0.05$，$p < 0.01$，$p < 0.005$ を示す．

これらから得られた中央値には，次のような傾向が見られた．

- ns

  Log4j，OpenJPA，Camel において中央値に差は見られず．HBase においては $\mathrm{DIFF}_{\mathrm{guru}}$ が $\mathrm{DIFF}_{\mathrm{szz}}$ の値を下回った．しかしながら，この差は小さく，また Log4j，HBase において有意差は存在しない．
- nd

  対象である全プロジェクトにおいて，中央値に差は見られず．また，OpenJPA, Camel においては有意差が存在しなかった．

表 6.1　Log4j における $\mathrm{DIFF}_{\mathrm{szz}}$ と $\mathrm{DIFF}_{\mathrm{guru}}$ の中央値と p 値およびその有意差

metric	$\mathrm{DIFF}_{\mathrm{szz}}$	$\mathrm{DIFF}_{\mathrm{guru}}$	p value
ns	1.00000	1.00000	4.79812e-01
nd	1.00000	1.00000	1.15135e-01
nf	2.00000	2.00000	3.48561e-01
entrophy	0.47948	0.83664	2.58344e-01
la	35.00000	39.00000	3.17999e-01
ld	4.50000	10.00000	7.71893e-02
lt	252.50000	253.43750	2.60420e-01
ndev	3.50000	8.00000	6.46460e-04***
age	7.58667	10.82199	1.19037e-01
nuc	22.00000	27.00000	4.42381e-01
exp	2509.50000	1243.00000	4.69491e-01
rexp	1.07154	1.08360	4.77138e-01
sexp	294.50000	609.00000	6.11542e-02
fix rate	15.00%	35.11%	

表 6.2　OpenJPA における $\mathrm{DIFF}_{\mathrm{szz}}$ と $\mathrm{DIFF}_{\mathrm{guru}}$ の中央値と p 値および有意差

metric	$\mathrm{DIFF}_{\mathrm{szz}}$	$\mathrm{DIFF}_{\mathrm{guru}}$	p value
ns	1.00000	1.00000	3.24992e-02*
nd	2.00000	2.00000	8.98472e-02
nf	3.00000	2.00000	1.51786e-01
entrophy	0.99836	0.83720	1.41704e-01
la	53.00000	41.50000	4.34386e-01
ld	6.00000	6.00000	4.77492e-01
lt	444.00000	414.12500	6.79530e-02
ndev	12.00000	15.50000	8.16886e-02
age	22.91326	13.63320	4.49405e-02*
nuc	34.00000	17.50000	1.09528e-04***
exp	894.00000	555.00000	4.73616e-04***
rexp	1.02498	1.04432	7.51753e-02
sexp	90.00000	94.50000	4.68559e-01
fix rate	12.79%	23.15%	

表 **6.3** **Camel** における $\mathrm{DIFF}_{\mathrm{szz}}$ と $\mathrm{DIFF}_{\mathrm{guru}}$ の中央値と **p** 値および有意差

metric	$\mathrm{DIFF}_{\mathrm{szz}}$	$\mathrm{DIFF}_{\mathrm{guru}}$	p value
ns	1.00000	1.00000	9.61711e-06***
nd	2.00000	2.00000	9.41452e-02
nf	3.00000	4.00000	2.51017e-06***
entrophy	1.31089	1.50977	4.74796e-05***
la	45.00000	87.00000	3.71714e-21***
ld	8.00000	8.00000	2.27471e-01
lt	153.50000	123.15530	1.11377e-11***
ndev	14.00000	25.00000	1.40597e-121***
age	11.91727	15.77485	8.82779e-04***
nuc	42.00000	29.00000	1.62257e-08***
exp	4976.50000	6215.75000	1.36397e-05***
rexp	1.05475	1.02112	5.76755e-11***
sexp	1203.00000	1261.50000	2.10397e-06***
fix rate	11.91%	17.85%	

表 **6.4** **HBase** における $\mathrm{DIFF}_{\mathrm{szz}}$ と $\mathrm{DIFF}_{\mathrm{guru}}$ の中央値と **p** 値および有意差

metric	$\mathrm{DIFF}_{\mathrm{szz}}$	$\mathrm{DIFF}_{\mathrm{guru}}$	p value
ns	2.00000	1.00000	2.02788e-01
nd	2.00000	2.00000	1.72945e-21***
nf	2.00000	3.00000	2.36854e-23***
entrophy	0.84535	1.14379	1.55122e-19***
la	23.00000	80.00000	5.91640e-52***
ld	6.00000	17.00000	9.23707e-30***
lt	824.75000	748.96667	4.59534e-02*
ndev	14.00000	35.00000	1.28355e-149***
age	7.05095	20.99476	1.74940e-32***
nuc	206.00000	111.00000	2.53801e-04***
exp	1558.25000	884.00000	3.06211e-09***
rexp	1.53296	1.03848	4.52753e-72***
sexp	729.50000	304.00000	8.74652e-17***
fix rate	14.83%	20.21%	

- nf

  中央値において，Log4j では同値が，OpenJPA では $DIFF_{szz}$ の方が大きい値が，Camel と HBase では $DIFF_{guru}$ の方が大きい値が得られた．しかしながら，この差は小さい．また，Log4j，OpenJPA において有意差は存在しなかった．

- entrophy

  OpenJPA においては $DIFF_{szz}$，Log4j，Camel，HBase においては $DIFF_{guru}$ の方が大きい値が得られた．また，有意差は Camel, HBase において存在する．

- la, ld

  OpenJPA においては $DIFF_{szz}$，Log4j，Camel，HBase においては $DIFF_{guru}$ の方が同値，または大きい値が得られた．また，有意差は Camel における la，HBase における la, ld に存在する．

- lt

  全プロジェクトを通じ，$DIFF_{guru}$, $DIFF_{szz}$ 間において値に大きな差はないものの，Log4j では $DIFF_{guru}$，OpenJPA，Camel，HBase においては $DIFF_{szz}$ の方が大きな値が得られた．また，有意差は Camel，HBase において存在する．

- ndev

  対象である全プロジェクトにおいて，$DIFF_{guru}$ の方が大きな値が得られた．また，有意差は Log4j，Camel，HBase において存在する．

- age

  OpenJPA では $DIFF_{guru}$，Log4j, Camel, HBase においては $DIFF_{szz}$ の方が小さな値が得られた．また，有意差は OpenJPA，Camel，HBase において存在する．

- nuc

  OpenJPA では $DIFF_{guru}$，Log4j, Camel, HBase においては $DIFF_{szz}$ の方が大きな値が得られた．また，有意差は OpenJPA，Camel，HBase において存在する．

- exp

  Camel では $DIFF_{szz}$，Log4j,OpenJPA，HBase においては $DIFF_{guru}$ の方が小さな値が得られた．また，有意差は OpenJPA，Camel，HBase において存在する．

- rexp

  Log4j，OpenJPA において，$DIFF_{szz}$ の方が値が小さいものの，その差は小さく，またそのに有意差は存在しなかった．これに対し，Camel，HBase においては

$DIFF_{guru}$ の方が小さい値が得られ，とりわけ HBase においてはその差も大きいものとなった．また，有意差は Camel, HBase において存在する．

- sexp

  HBase では $DIFF_{guru}$，Log4j, OpenJPA，Camel では $DIFF_{szz}$ の方が小さな値が得られた．また，有意差は Camel, HBase において存在する．

- fix rate

  対象である全プロジェクトにおいて，$DIFF_{guru}$ の方が高い確率となった．

$DIFF_{szz}$ は，自身が持つ中央値の大小から $DIFF_{guru}$ と比べると Buggy である可能性が高いと考えられるメトリクスにおいて，有意差のある値を持っていない傾向が見られる．これに対し，$DIFF_{guru}$ は，$DIFF_{szz}$ と比べ Buggy である可能性が高いと考えられるメトリクスに対し，そのほとんどで有意差が存在する値を持っている．

また，6.1 で述べたように，メトリクスの値はその値に対する明確な基準値というものは持っていないながらも，その大小は不具合コミットの推定結果の判断基準にある程度寄与すると考えられ，先行研究において，オープンソースプロジェクトを対象とした不具合コミット推定の可否判断に有用であるメトリクスとして，nf, la, fix, age が挙げられている [8, 20]．

la, fix においては $DIFF_{guru}$ が値が高く, Buggy である可能性が高いことが考えられる．nf においては，$DIFF_{guru}$ が $DIFF_{szz}$ を上回っているプロジェクトにのみ有意差が存在する．age においては，全て $DIFF_{szz}$ の方が値が小さい．しかしながら，5.2.1 節において述べたように，Buggy，Clean と推定されるコミット間でも両手法において Buggy であると推定されたコミットの方が値が大きかったことから，age におけるデータ比較への妥当性については疑念の残るところである．

また $DIFF_{szz}$，$DIFF_{guru}$ の値の分布を箱ひげ図を用いて図 6.1〜図 6.51 に示す．

ほとんどの箱ひげ図において，$DIFF_{guru}$ の方が外れ値が明らかに多く，第一四分位点，第三四分位点の間が大きく開いていることから，$DIFF_{guru}$ におけるデータ群はまとまりがないことが考えられる．Clean，ないしは Buggy と推定されるコミット間においては何らかの共通の特徴が現われると考えられ，$DIFF_{szz}$ は $DIFF_{guru}$ よりも Buggy であるコミットが含まれている可能性は低いと考えられるため，$DIFF_{guru}$ に Buggy と推定されるコミットが含まれている可能性は否定できない．

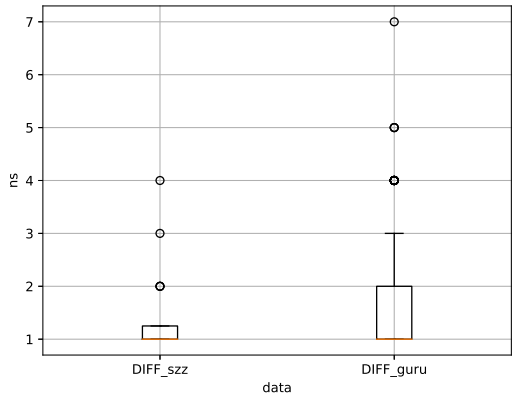

図 6.1　Log4j における ns の分布

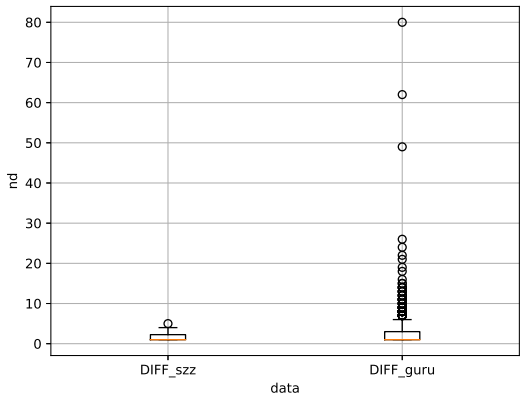

図 6.2　Log4j における nd の分布

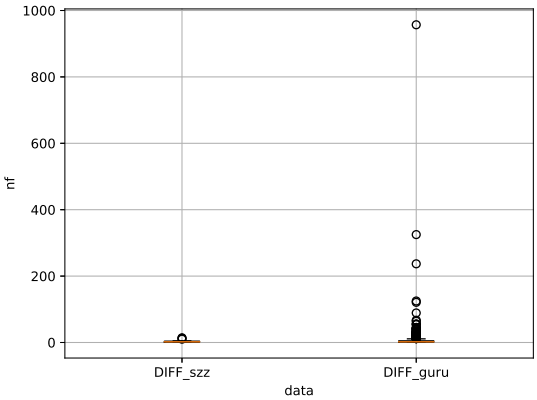

図 6.3　Log4j における nf の分布

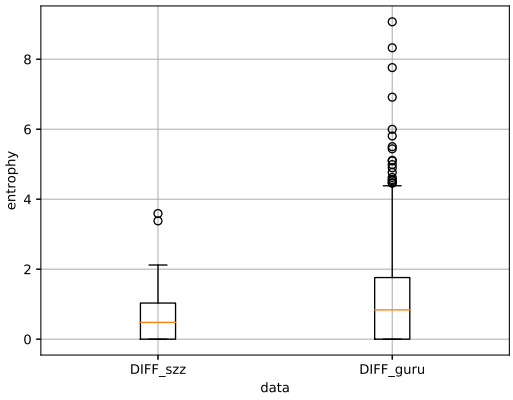

図 6.4　Log4j における entrophy の分布

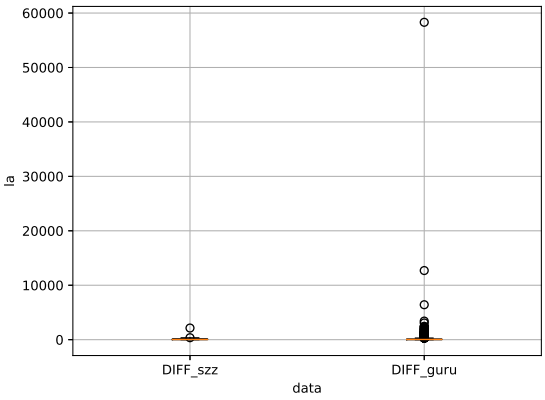

図 6.5　Log4j における la の分布

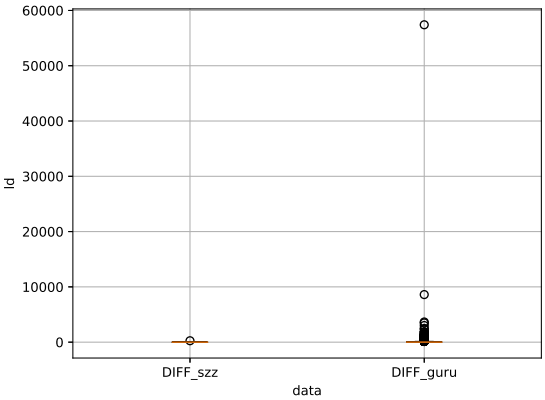

図 6.6　Log4j における ld の分布

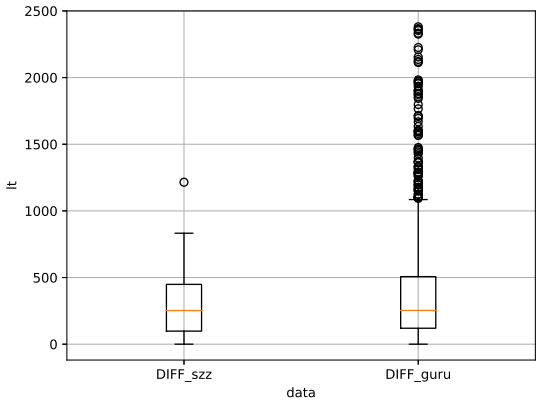

図 6.7　Log4j における lt の分布

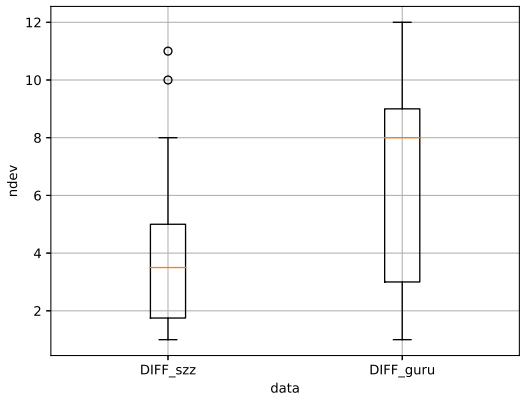

図 6.8　Log4j における ndev の分布

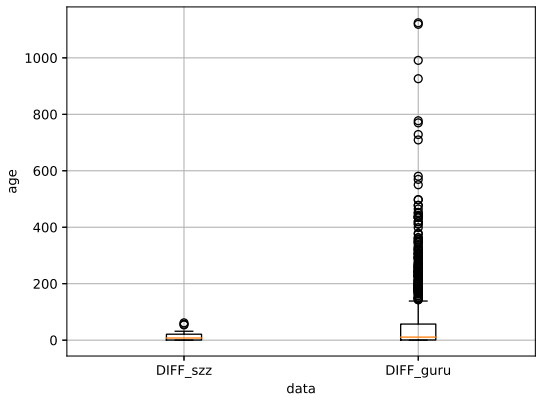

図 6.9　Log4j における age の分布

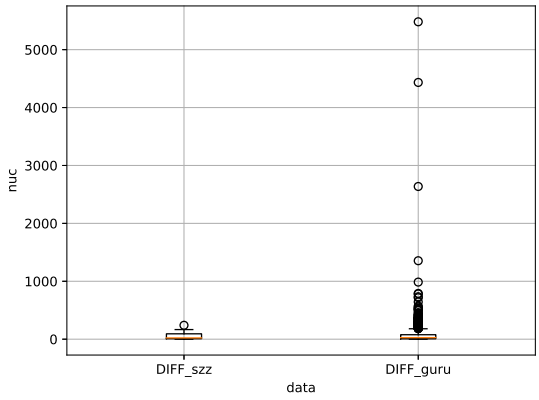

図 6.10　Log4j における nuc の分布

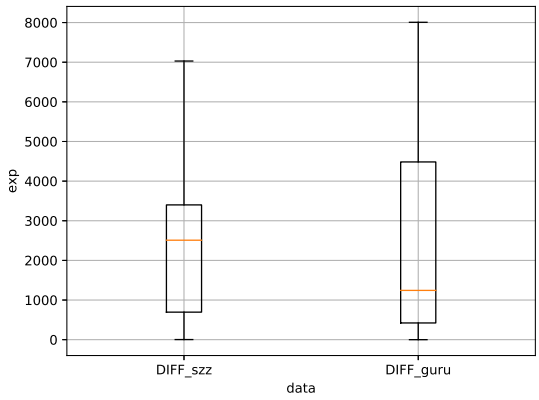

図 6.11　Log4j における exp の分布

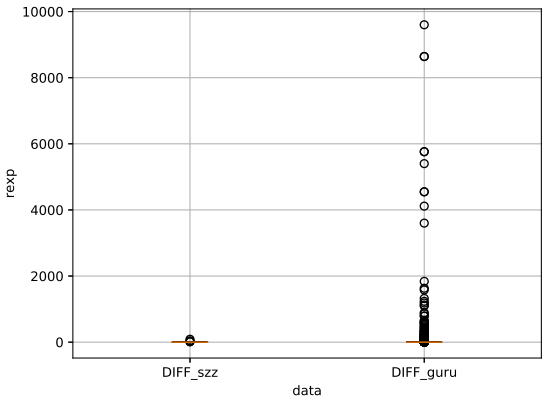

図 6.12　Log4j における rexp の分布

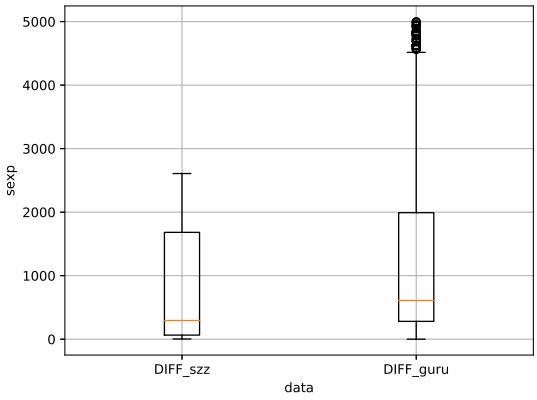

図 6.13　Log4j における sexp の分布

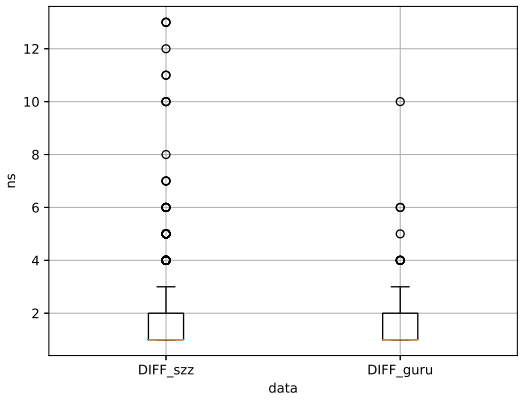

図 6.14　OpenJPA における ns の分布

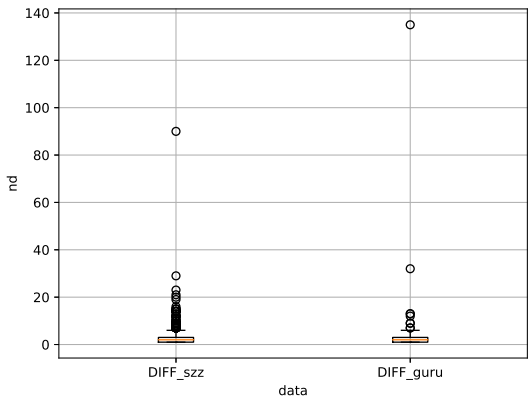

図 6.15　OpenJPA における nd の分布

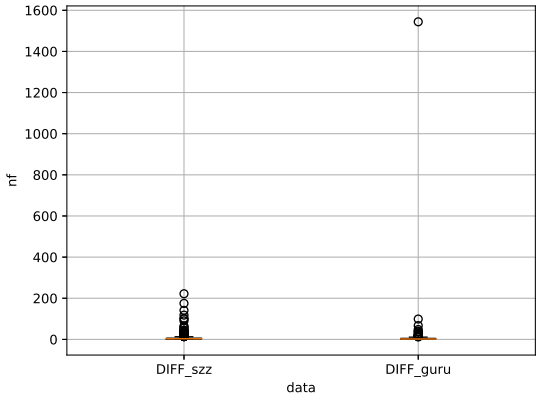

図 6.16　OpenJPA における nf の分布

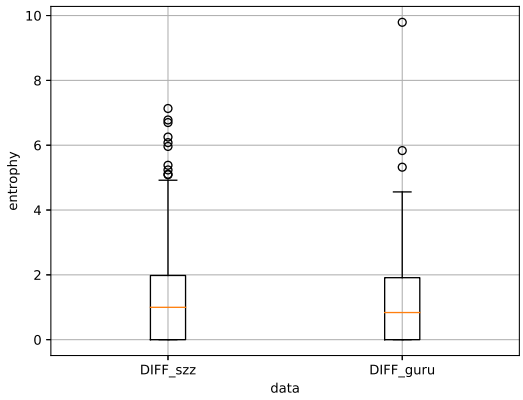

図 6.17　OpenJPA における entrophy の分布

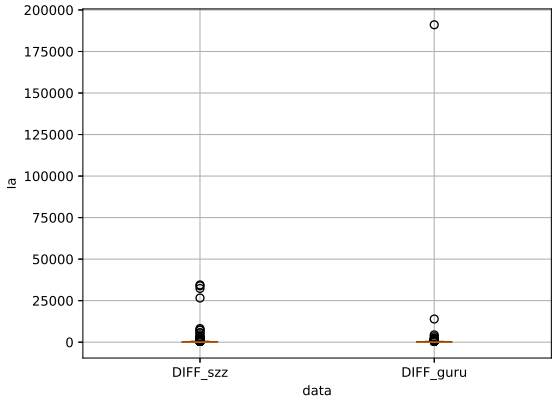

図 6.18　OpenJPA における la の分布

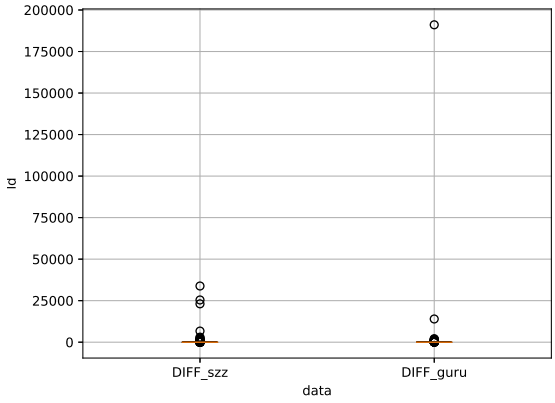

図 6.19　OpenJPA における ld の分布

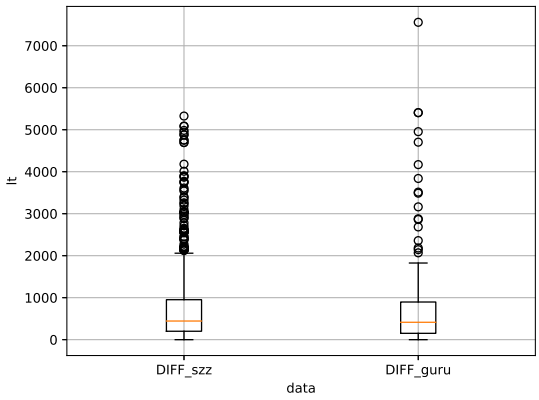

図 6.20　OpenJPA における lt の分布

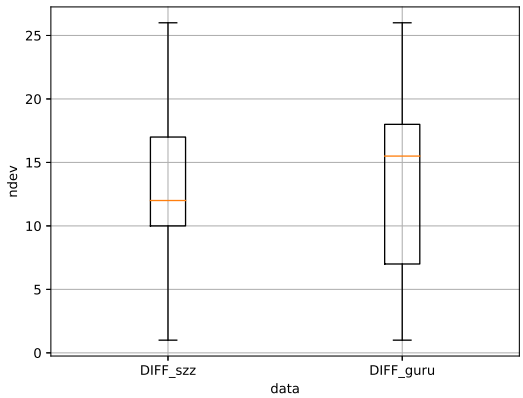

図 6.21　OpenJPA における ndev の分布

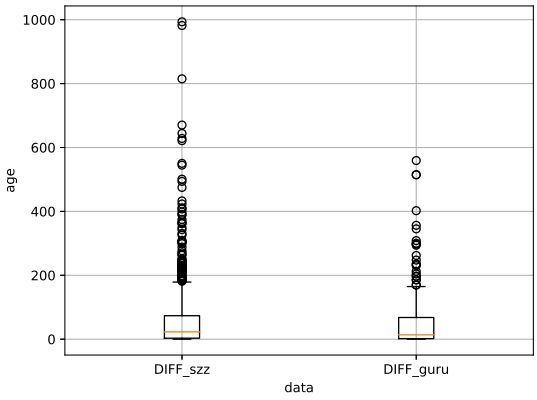

図 6.22　OpenJPA における age の分布

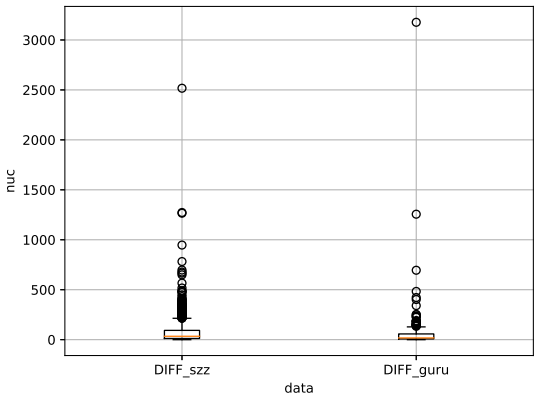

図 6.23　OpenJPA における nuc の分布

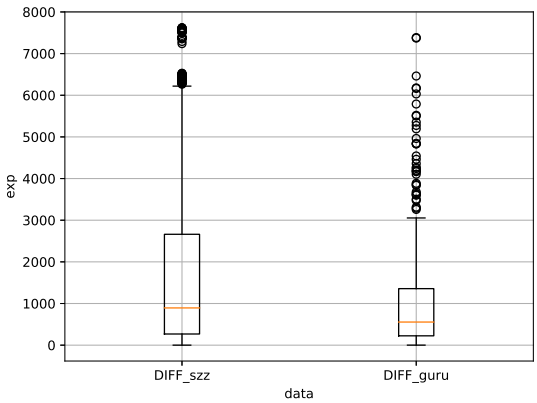

図 6.24　OpenJPA における exp の分布

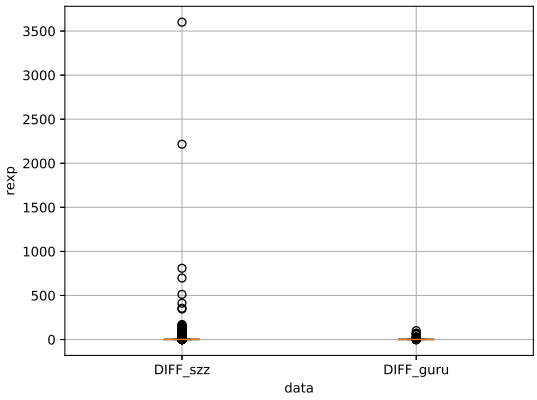

図 6.25　OpenJPA における rexp の分布

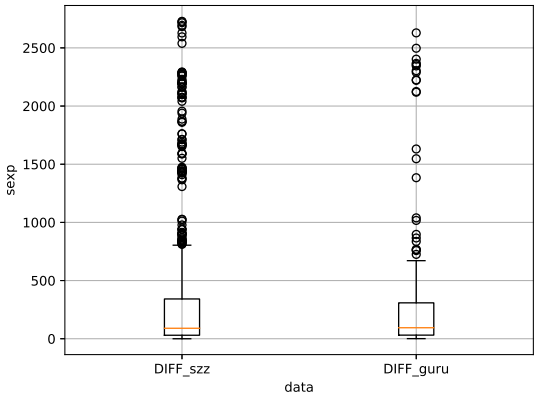

図 6.26　OpenJPA における sexp の分布

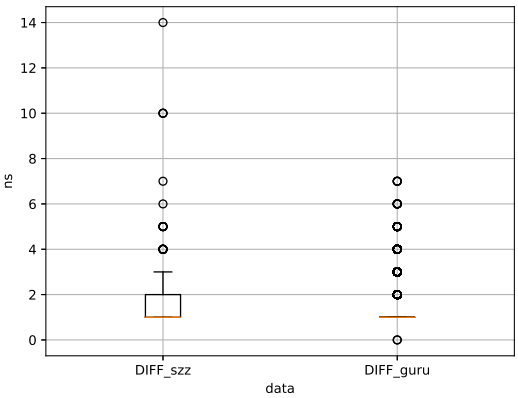

図 6.27　Camel における ns の分布

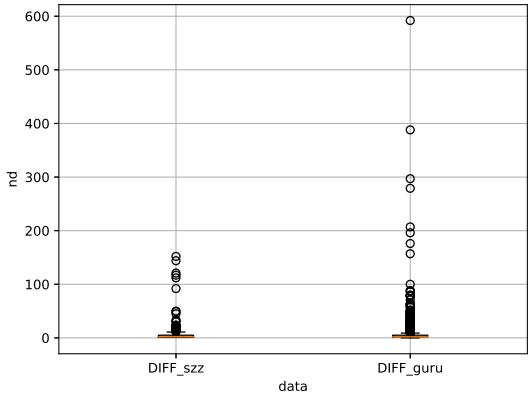

図 6.28　Camel における nd の分布

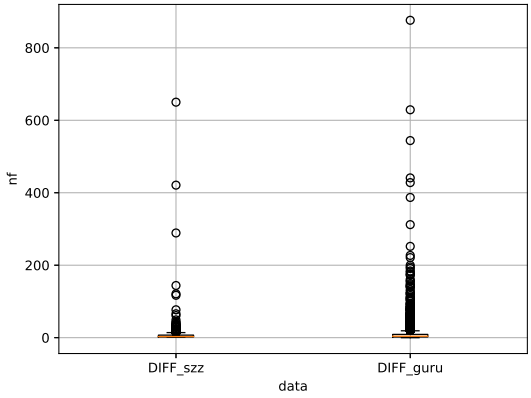

図 6.29　Camel における nf の分布

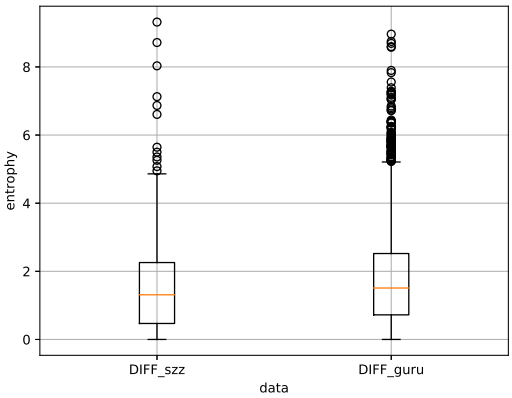

図 6.30　Camel における entrophy の分布

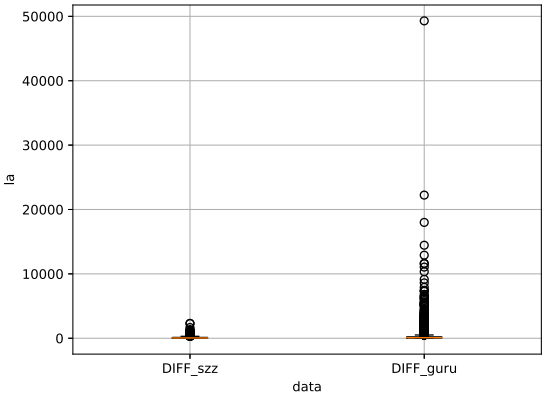

図 6.31　Camel における la の分布

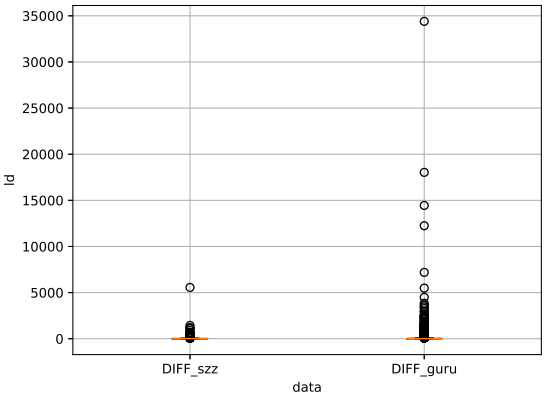

図 6.32　Camel における ld の分布

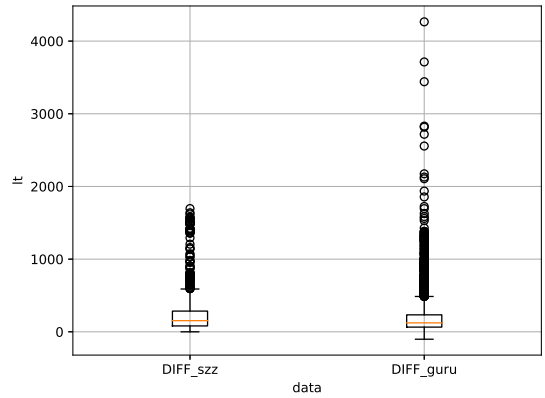

図 6.33　Camel における lt の分布

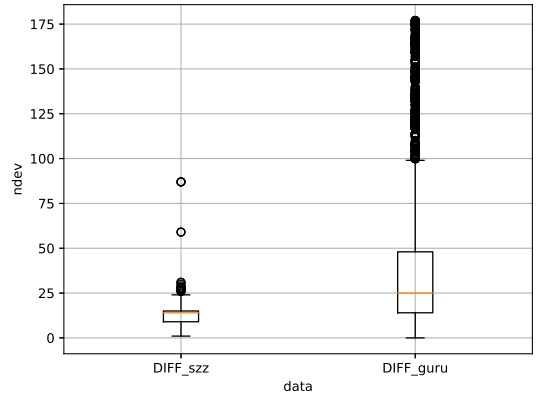

図 6.34　Camel における ndev の分布

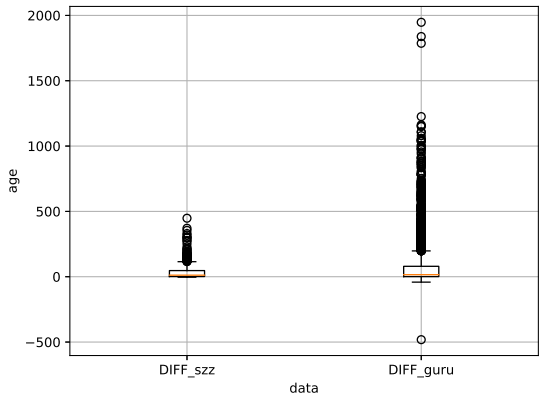

図 6.35　Camel における age の分布

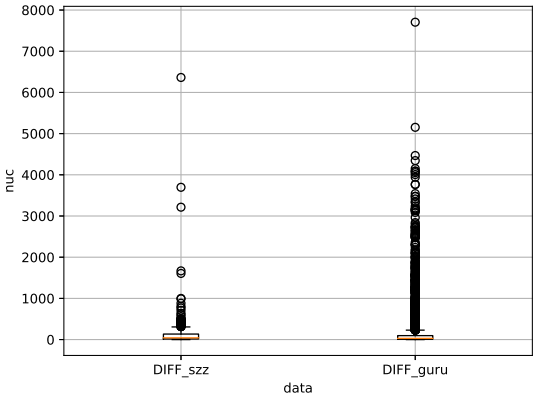

図 6.36　Camel における nuc の分布

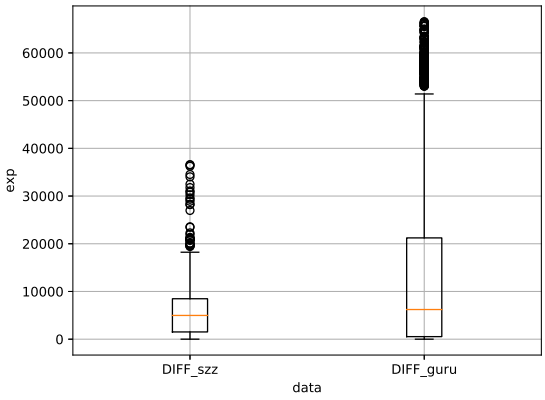

図 6.37　Camel における exp の分布

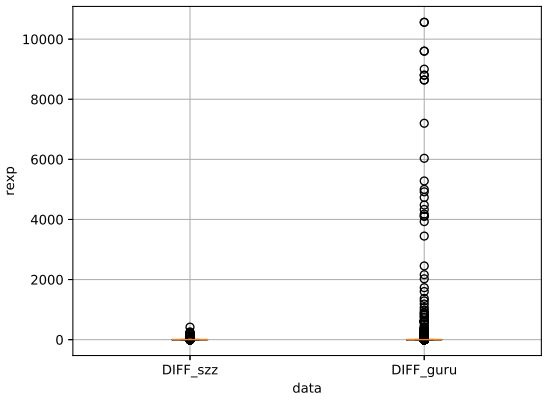

図 6.38　Camel における rexp の分布

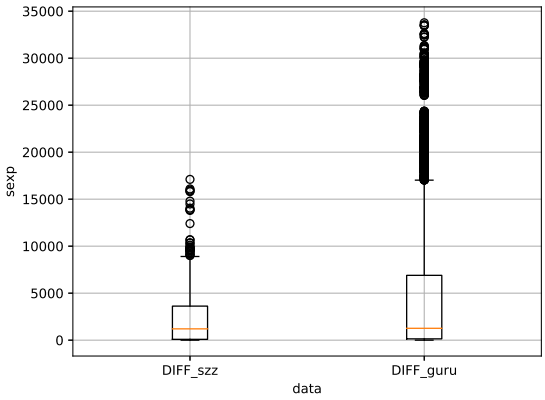

図 6.39　Camel における sexp の分布

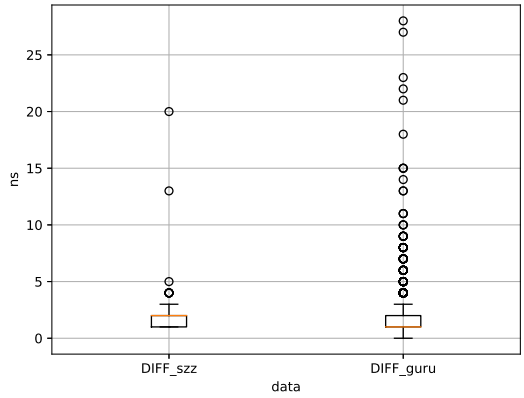

図 6.40　HBase における ns の分布

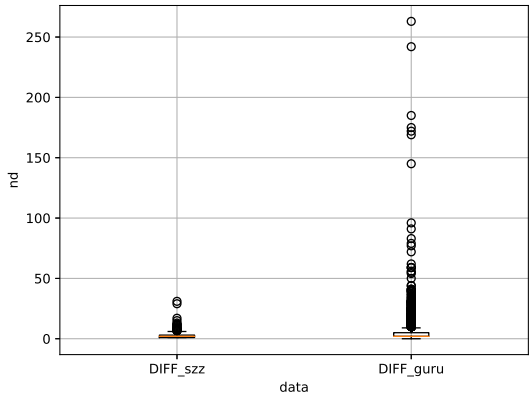

図 6.41　HBase における nd の分布

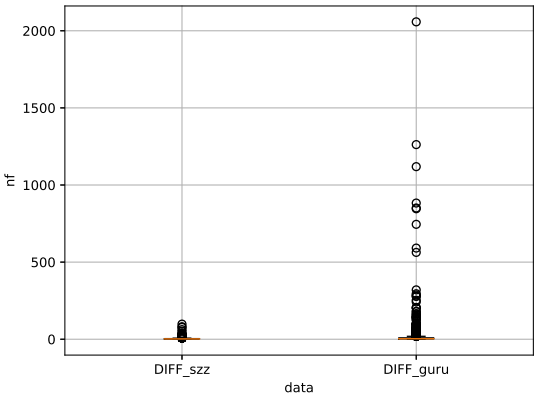

図 6.42　HBase における nf の分布

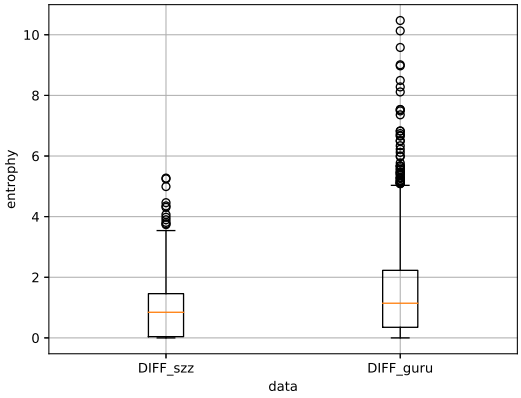

図 6.43　HBase における entrophy の分布

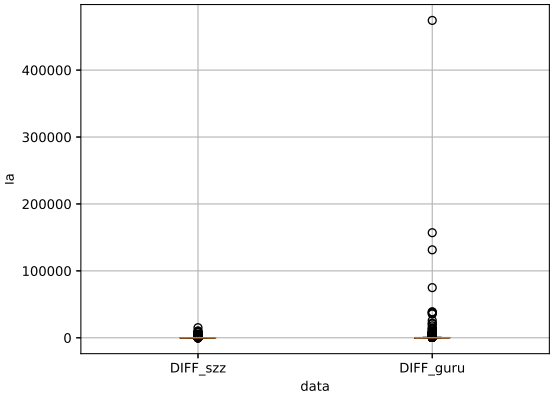

図 6.44　HBase における la の分布

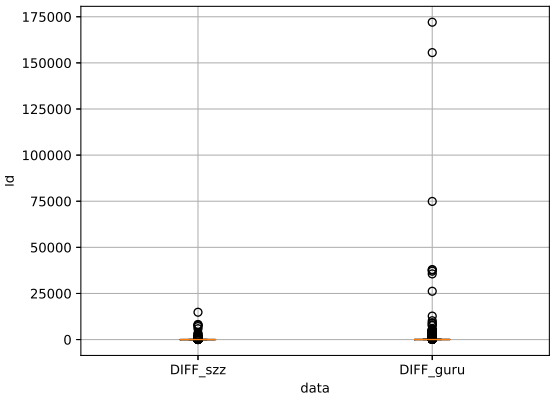

図 6.45　HBase における ld の分布

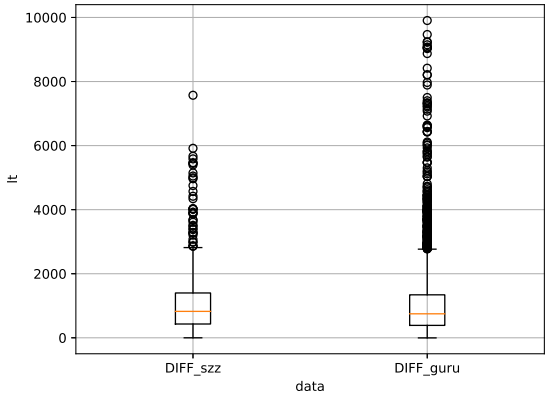

図 6.46　HBase における lt の分布

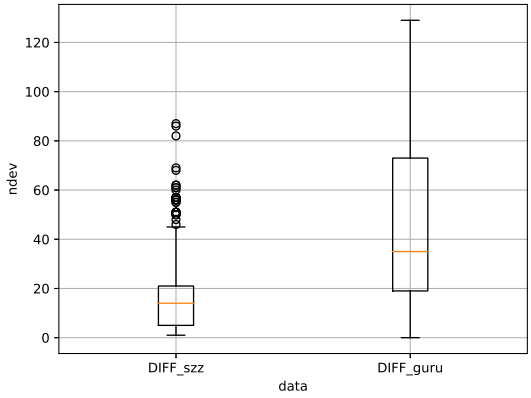

図 6.47　HBase における ndev の分布

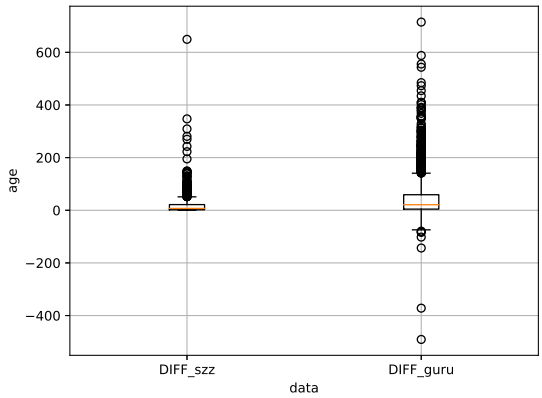

図 6.48　HBase における age の分布

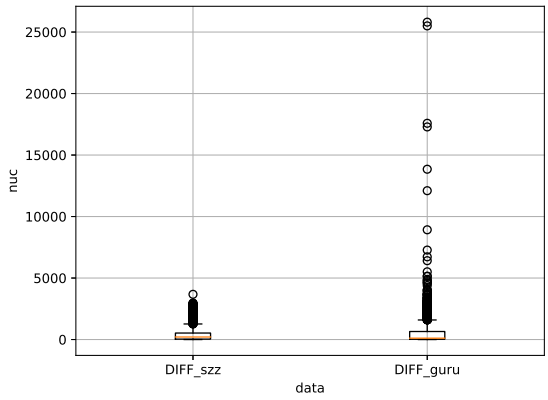

図 6.49　HBase における nuc の分布

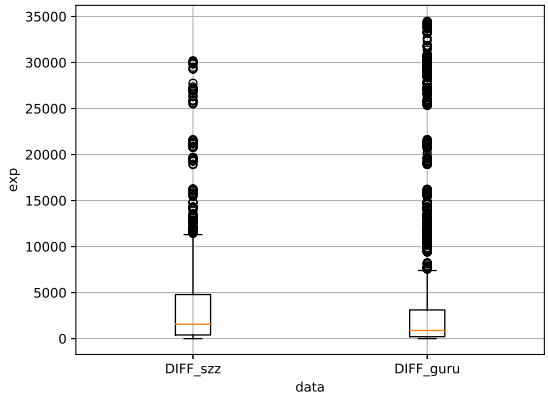

図 6.50　HBase における exp の分布

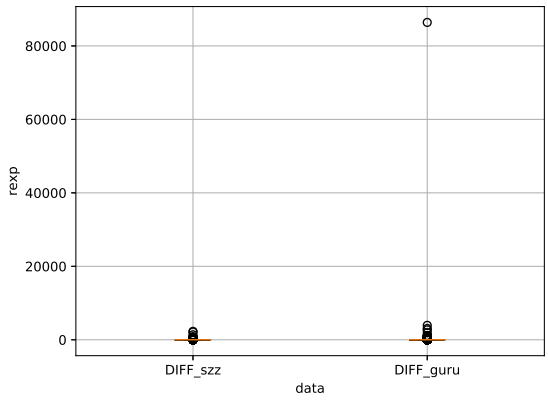

図 6.51　HBase における rexp の分布

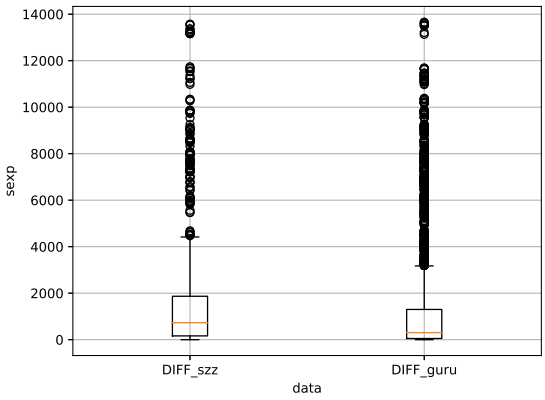

図 6.52　HBase における sexp の分布

これを踏まえ，RQ2 の回答を行う．

先行研究 [8, 20] において不具合コミット推定の可否判断に有用であるとされたメトリクスに対し，$\mathrm{DIFF}_{\mathrm{guru}}$ はある程度の Buggy である可能性が高いという考えに即した値を持っている．また，有用ではないとされた他のメトリクスのいくつかにおいても同様である．これに対し，$\mathrm{DIFF}_{\mathrm{szz}}$ では，不具合コミット推定の可否判断に有用であるとされたメトリクスに対し，Buggy である可能性が高いという考えに即した値はあまり存在せず，即した値を持っている場合においても，その有意差は存在するとは言い難い．また，$\mathrm{DIFF}_{\mathrm{guru}}$ のデータ分布にはまとまりがないことから，$\mathrm{DIFF}_{\mathrm{guru}}$ において Buggy であると考えられるコミットが含まれている可能性は高いものと思われる．

これより，$\mathrm{DIFF}_{\mathrm{guru}}$，$\mathrm{DIFF}_{\mathrm{szz}}$ の間には何らかの差が存在する可能性が高く，その差として，$\mathrm{DIFF}_{\mathrm{guru}}$ には Buggy である確率が高いコミットが存在する可能性が考えられる．

また，RQ1 および RQ2 の回答から，Commit Guru のデータを正解データとしての可能性も存在すると考えられる．

7. 結言

ソフトウェアの複雑さと重要性が日々増してきている昨今，ソフトウェアの品質予測は重要な研究テーマであり，その研究は活発化している．その中で，不具合の正解データとして用いられることの多い Commit Guru のソフトウェア不具合コミットの情報に対する信頼性は不確かなものであった．そこで，本研究では Commit Guru のソフトウェア不具合コミットの情報における正解データとしての信頼性検証のため，ソフトウェア不具合コミット推定手法である SZZ アルゴリズムとの結果の整合性を比較し，考察を行なった．その結果，Commit Guru のソフトウェア不具合コミットの情報は不具合の正解データとして可能性を示すものとなった．

謝辞

本研究を行うにあたり，研究課題の設定や研究に対する姿勢，本報告書の作成に至るまで，全ての面で丁寧なご指導を頂きました，本学情報工学・人間科学系水野修教授に厚く御礼申し上げます．本報告書執筆にあたり貴重な助言を多数頂きました，本学ソフトウェア工学研究室の皆さん，本学生命物質科学域応用生物学課程 國領正真君，学生生活を通じて著者の支えとなった家族や友人に深く感謝致します．

参考文献

[1] G. Tassey, “The economic impacts of inadequate infrastructure for software testing,” 2002.

[2] E. Shihub, “An exploration of challenges limiting pragmatic software defect prediction,” 2012.

[3] J. Śliwerski, T. Zimmermann, and A. Zeller, “When do changes induce fixes?,” Proceedings of the 2005 International Workshop on Mining Software Repositories, pp.1–5, Proc. of 2nd International workshop on Mining software repositories, ACM, New York, NY, USA, 2005.

[4] C. Rosen, B. Grawi, and E. Shihab, “Commit guru: Analytics and risk prediction of software commits,” Proceedings of the 2015 10th Joint Meeting on Foundations of Software Engineering, pp.966–969, ESEC/FSE 2015, ACM, New York, NY, USA, 2015.

[5] E. Shihub, B. Grawi, C. Rosen, and S. Wehaibi, Commit Guru, （オンライン）, 入手先〈http://commit.guru〉.

[6] Bugzilla, （オンライン）, 入手先〈https://www.bugzilla.org〉.

[7] Apache’s JIRA issue tracker, （オンライン）, 入手先〈https://issues.apache.org/jira/secure/Dashboard.jspa〉.

[8] Y. Kamei, E. Shihab, B. Adams, A.E. Hassan, A. Mockus, A. Sinha, and N. Ubayashi, “A large-scale empirical study of just-in-time quality assurance,” IEEE Trans. Softw. Eng., vol.39, no.6, pp.757–773, June 2013.

[9] A. Hindle, D.M. German, and R. Holt, “What do large commits tell us?: A taxonomical study of large commits,” Proceedings of the 2008 International Working Conference on Mining Software Repositories, pp.99–108, Proc. of 5th International workshop on Mining software repositories, ACM, New York, NY, USA, 2008.

[10] A. Mockus and L.G. Votta, “Identifying reasons for software changes using historic databases,” Proceedings of the International Conference on Software Maintenance

(ICSM'00), pp.120–130, ICSM '00, IEEE Computer Society, Washington, DC, USA, 2000.

[11] A. Mockus and D.M. Weiss, "Predicting risk of software changes," Bell Labs Technical Journal, vol.5, no.2, pp.169–180, aug 2002.

[12] S. Matsumoto, Y. Kamei, A. Monden, K.-i. Matsumoto, and M. Nakamura, "An analysis of developer metrics for fault prediction," Proceedings of the 6th International Conference on Predictive Models in Software Engineering, pp.18:1–18:9, PROMISE '10, ACM, New York, NY, USA, 2010.

[13] N. Nagappan, T. Ball, and A. Zeller, "Mining metrics to predict component failures," Proceedings of the 28th International Conference on Software Engineering, pp.452–461, Proc. of 28th International Conference on Software Engineering, ACM, New York, NY, USA, 2006.

[14] M. D'Ambros, M. Lanza, and R. Robbes, "An extensive comparison of bug prediction approaches," 2010 7th IEEE Working Conference on Mining Software Repositories (MSR 2010), pp.31–41, 2010.

[15] A.E. Hassan, "Predicting faults using the complexity of code changes," Proceedings of the 31st International Conference on Software Engineering, pp.78–88, Proc. of 31st International Conference on Software Engineering, IEEE Computer Society, Washington, DC, USA, 2009.

[16] R. Moser, W. Pedrycz, and G. Succi, "A comparative analysis of the efficiency of change metrics and static code attributes for defect prediction," Proceedings of the 30th International Conference on Software Engineering, pp.181–190, Proc. of 30th International Conference on Software Engineering, ACM, New York, NY, USA, 2008.

[17] N. Nagappan and T. Ball, "Use of relative code churn measures to predict system defect density," Proceedings of the 27th International Conference on Software Engineering, pp.284–292, ACM, New York, NY, USA, 2005.

[18] A.G. Koru, D. Zhang, K.E. Emam, and H. Liu, "An investigation into the functional form of the size-defect relationship for software modules," IEEE Transactions on

Software Engineering, vol.35, pp.293–304, 2009.

[19] T.L. Graves, A.F. Karr, J.S. Marron, and H.P. Siy, "Predicting fault incidence using software change history," IEEE Trans. Software Eng., vol.26, pp.653–661, 2000.

[20] E. Shihab, A.E. Hassan, B. Adams, and Z.M. Jiang, "An industrial study on the risk of software changes," Proceedings of the ACM SIGSOFT 20th International Symposium on the Foundations of Software Engineering, pp.62:1–62:11, FSE '12, ACM, New York, NY, USA, 2012.