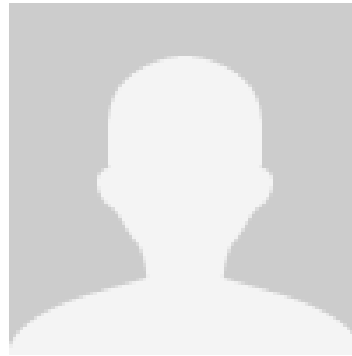


A Comparative Study of Bug Patterns in Java Cloned and Non-cloned Code



Hitesh Sajnani



Vaibhav Saini



Cristina V. Lopes

Bren School of Information and Computer Sciences
University of California, Irvine, Irvine, CA, USA

コードクローンとFindBugsによって検出されてたバグパターンとの関係性を調べるために、31のオープンソースJavaプロジェクトに対して調査を行った

INTRODUCTION：背景

- 最近の開発ではコピーアンドペーストによるコードの再利用が一般的な手法となっている
- これまではコードクローンは有害なものとして考えられてきた
- 最近の研究では、クローンの有害な性質について疑問視するような研究結果が出ている
- ある研究では「ソフトウェア品質に与える影響を調べるには情報が少なすぎる」と言われている

INTRODUCTION：目的

- 本研究ではコードクローンとバグの関係を調べるために31のオープンソースJavaプロジェクトを対象とした調査を行う
 - FindBugsが検出するバグパターンとコードクローンの関係を調べる
- 今回行う分析はコード品質とクローン作業の関連を調査するのに役に立ち，この分野での研究に役立つと考えている

Research Question

- RQ1: クローンされたコードの欠陥密度はクローンされていないコードよりも大きいのか？
 - 欠陥密度：1000行ごとのFindBugsで検出されたバグパターン数
- RQ2: クローンコードで頻繁にみられる特定のバグカテゴリは存在するか？
- RQ3: クローン作業を通じたバグパターンの広がりはどのくらいの頻度で起きるか？

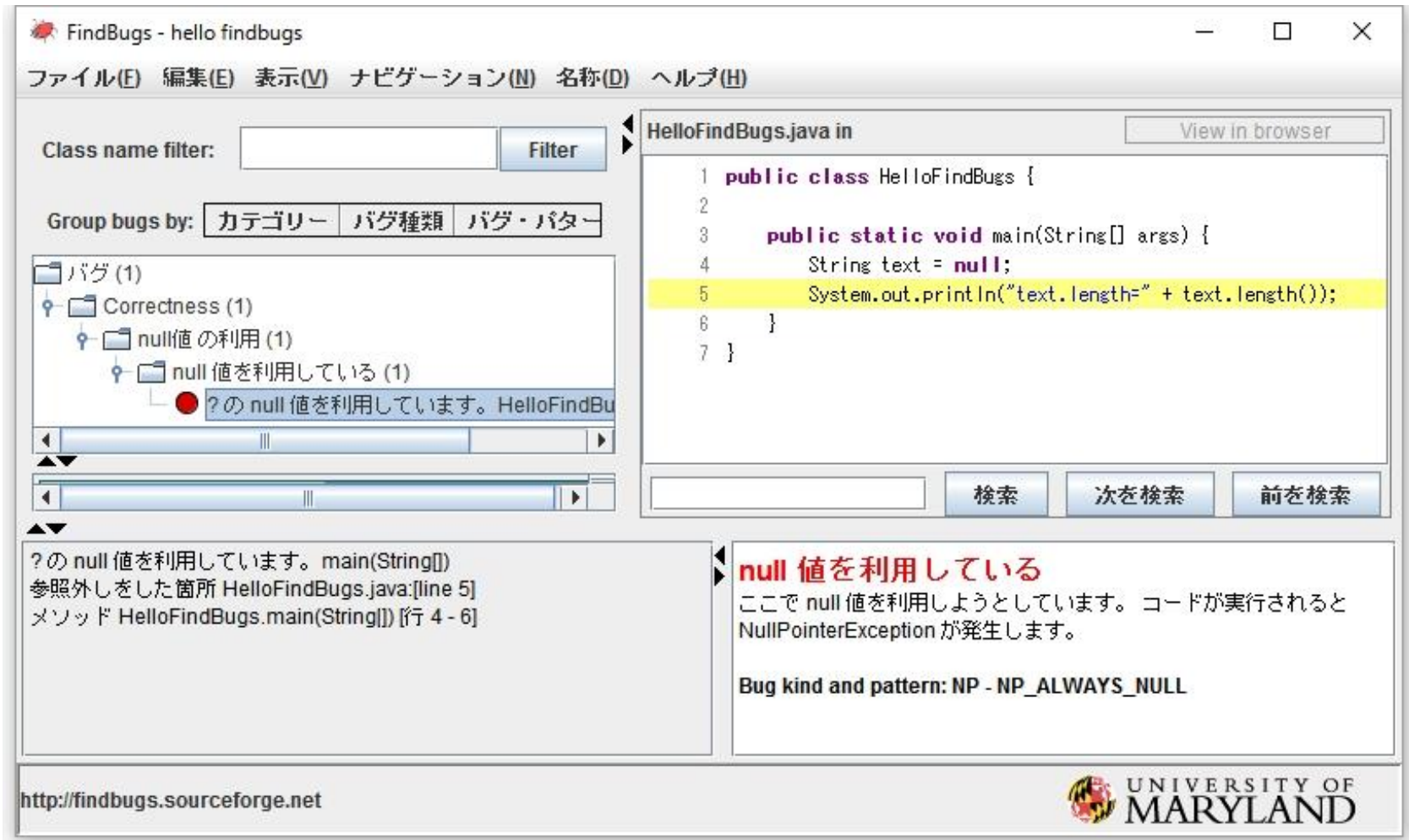
FindBugs

- Javaのソースコードを静的に解析して，バグとなりそうなコードを見つけるツール
 - バグカテゴリ：
 - CORRECTNESS, SECURITY, BAD_PRACTICEなど
 - バグパターン（例）：
 - IL_INFINITE_LOOP（無限ループ）
 - XSS_REQUEST_PARAMETER_TO_JSP_WRITER（XSS脆弱性のあるJSP）
 - ES_COMPARING_PARAMETER_STRING_WITH_EQ（Stringを=や!=を使って比較）

FindBugs：カテゴリー一覧

バグカテゴリ	説明
CORRECTNESS	バグの可能性—明らかなコーディングミスによって、開発者の意図しないコードになっている
SECURITY	リモートから悪用可能なセキュリティ上の脆弱性につながる、信頼されていない入力方法
BAD_PRACTICE	推奨される、あるいは本質的なコーディング規範からの逸脱。例えば、hashCodeとequalsの問題、cloneableイディオム、例外の無視、直列化の問題、finalizeの誤用といったもの。開発グループによっては、これらの幾つかの規範を気にとめないかもしれないが、できる限り正確な解析を行う必要がある。
STYLE	誤解を与え易いコード、変則的、間違いを招き易いコード。例えば不要なローカル変数への格納、switchのフォールスルー、確認されていないキャスト、nullであることが明らかな値に対する冗長なnullチェックなど。幾らかの誤検出が許容される。
PERFORMANCE	必ずしも間違いではないが、実行効率が悪くなる可能性のあるもの。
MALICIOUS_CODE	信頼されていないコードからの攻撃に対して脆弱性を持つコード
MT_CORRECTNESS	スレッド、ロック、volatileの扱いに対するコードの欠陥
I18N	国際化、ロカールの扱いに対するコードの欠陥

FindBugs : 使用例



STUDY DESIGN

- First Step
 - クローン検出ツールを用いて，メソッド単位のクローンを検出対象としてクローンを特定する
- Second Step
 - FindBugsでバグパターンとバグカテゴリを検出し，メソッドとバグパターンの関連付け，メソッドとバグカテゴリの関連付けを行う
- これによって得られるメソッド単位のクローンとメソッドに関連するバグパターン，バグカテゴリを用いてRQに解答する

Subject Systems, Clone Detection

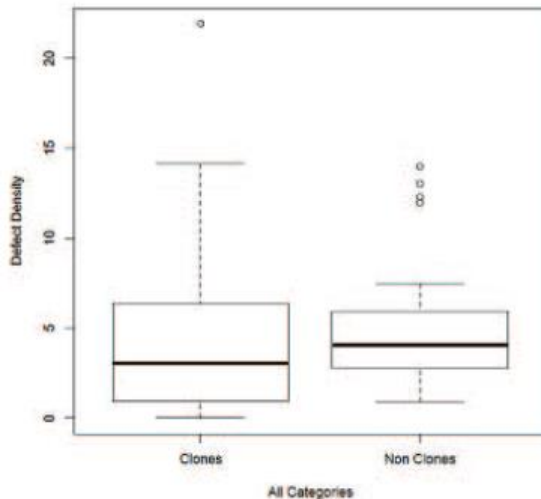
- 対象プロジェクト：31プロジェクト
 - ApacheのJavaプロジェクト
 - 検索，データベース，サーバ，分散システム，機械学習，自然言語処理など，分野は様々で，規模も様々である
- クローン検出方法
 - トークンベースのクローン検出ツール
 - 著者らが作成した，メソッド単位でタイプ3のクローンを検出することができるツール

Why use Bug Patterns ?

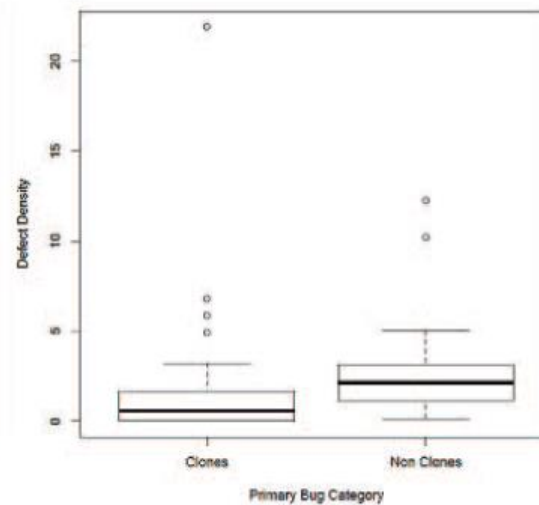
- 品質の調査方法として、Bug Reportと静的ツールを用いる方法が考えれるがFindBugsを選んだ理由
 - BTSとコードリポジトリが必ずしもリンクしているわけではない
 - BTSではReportが誤分類されていることがある
 - 人気のあるプロジェクトとそうでないプロジェクトではバグ報告数に違いがある
 - バグ報告数はプロジェクトの期間に比例するので、開始したばかりのプロジェクトと長期間存在するプロジェクトを比較できない

STUDY RESULTS : RQ1

- RQ1: クローンされたコードの欠陥密度はクローンされていないコードよりも大きいのか？

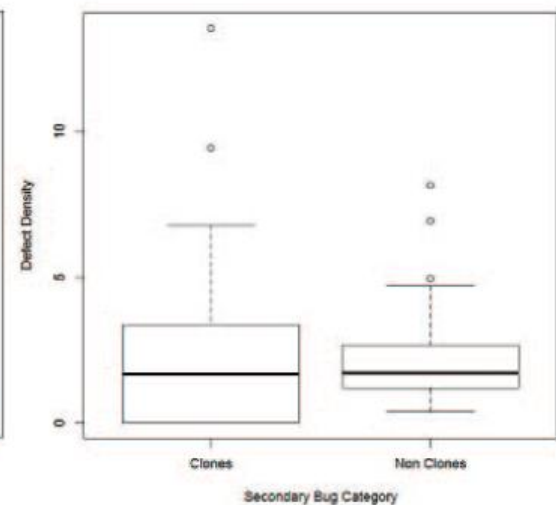


All Categories :



Primary Categories :

Correctness
Multi-threaded Correctness
Performance
Security

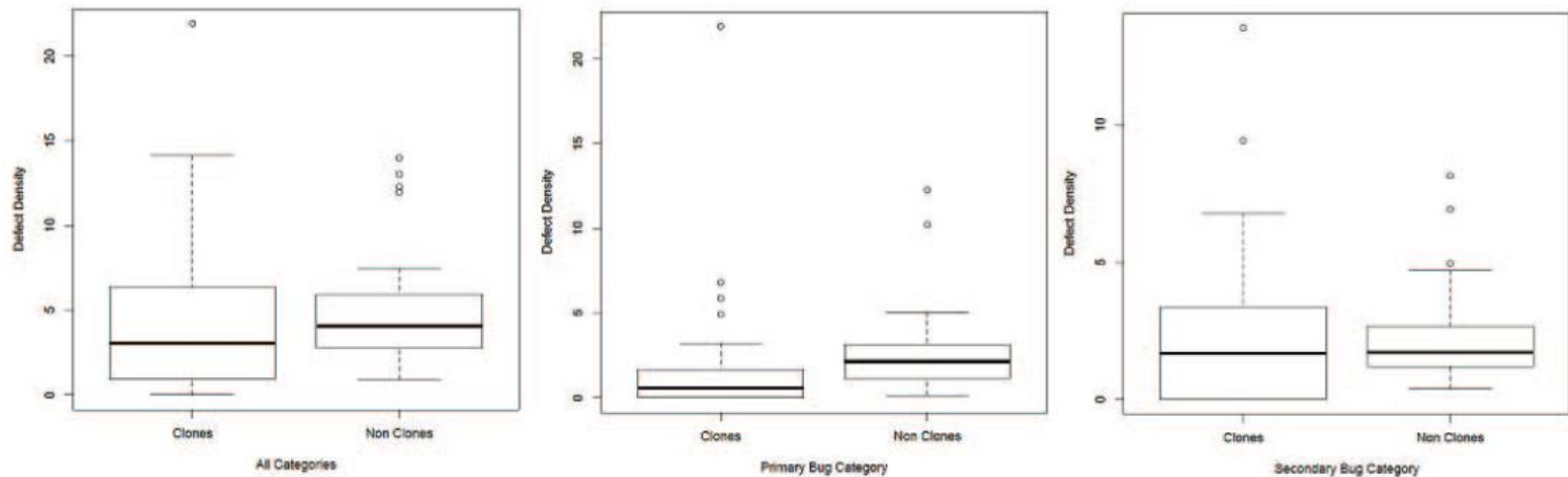


Secondary Categories :

Bad Practice
Style

STUDY RESULTS : RQ1

- RQ1: クローンされたコードの欠陥密度はクローンされていないコードよりも大きいのか？



FindBugsを用いて計算された欠陥密度(Primary Categories)はクローンでないコードがクローンコードに比べて3.7倍多い

STUDY RESULTS : RQ2

- RQ2: クローンコードで頻繁にみられる特定のバグカテゴリは存在するか？

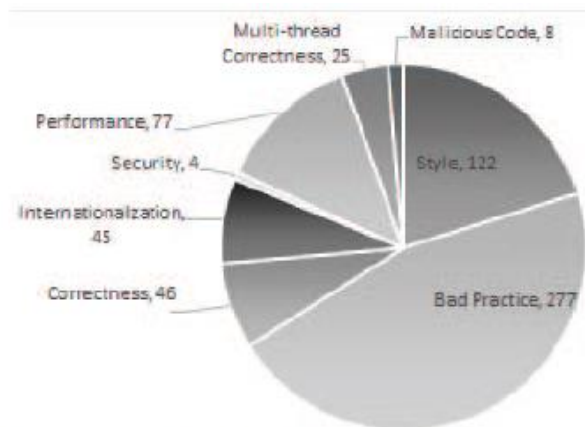


Fig. 4: Bug patterns in cloned code classified into various categories

コードクローンに関連するバグパターンの66%はFindBugsカテゴリの中で最も問題のない2つであるSTYLEとBad Practiceである。クローンでないコードでは49%である

STUDY RESULTS : RQ3

- RQ3: クローン作業を通したバグパターンの広がり
りはどのくらいの頻度で起きるか？
 - 339のコードクローンを目視してバグパターンを確認
 - 2つのカテゴリに分ける
 - Category 1: クローンを構成するコードの複数に重複するバグパターンがある
 - Category 2: クローンを構成するコードの一つにのみバグパターンがある場合

75%のFindBugsのバグパターンを含むクローンは変更されることなくそのまま利用されていた

CONCLUSION

- 本研究によって、クローンが必ずしも有害なものではないことがわかりました。
- 本研究を含むクローンの研究では、コードクローンにはいくつかの長所と短所があることを示唆する結果がある
- 今後は開発者がクローンを効果的に管理してクローン作業の短所の部分に対処することができるツールやテクニックに焦点が当たるかもしれない

所見

- FindBugsを使えば，Bugの分類などがきっちりできるのでやりやすそうに思えた．
- FindBugsでは見つからないようなBugとの関連を見ることが必要だと思った．
- クローンの種類（コードの自動生成，APIやLibraryのプロトコルに依存したことによって作成されるクローンなど）を見れたのが良かった