

Understanding Code Smell Detection via Code Review: A Study of the OpenStack Community

Xiaofeng Han¹, Amjed Tahir², Peng Liang^{1*}, Steve Counsell³, Yajing Luo¹

¹School of Computer Science, Wuhan University, Wuhan, China

²School of Fundamental Sciences, Massey University, Palmerston North, New Zealand

³Department of Computer Science, Brunel University London, London, United Kingdom hanxiaofeng@whu.edu.cn,
a.tahir@massey.ac.nz, liangp@whu.edu.cn, steve.counsell@brunel.ac.uk, luoyajing@whu.edu.cn

*2021 IEEE/ACM 29th International Conference on
Program Comprehension (ICPC), 2021, pp. 323-334,*

背景

- コードスメル自動検出手法が数多く研究、開発されてきた
- 偽陽性が多い、ドメイン固有問題を考慮しないなどの問題がある
- コードレビューによる検出が有効と考えらるが、どれ程効果的に検出、除去されているか不明

目的, アプローチ

- 目的

- コードレビューでのコードスメル検出の有効性を明らかにする
- コードレビューでコードスメルがどう対処されているかを明らかにする

- アプローチ

- OpenStackで活発なプロジェクトからコードスメルに関連するレビューを特定し, 分析

リサーチクエスト

1. 頻繁に検出されるコードスメルは何か
2. コードスメルの発生原因は何か
3. レビューアー, プログラマはコードスメルにどう対処するか
 1. レビューアーはどのような提案をするか
 2. プログラマはどのような修正を加えるか
 3. レビューアーの提案とプログラマの修正にどのような関係があるか

分析概要

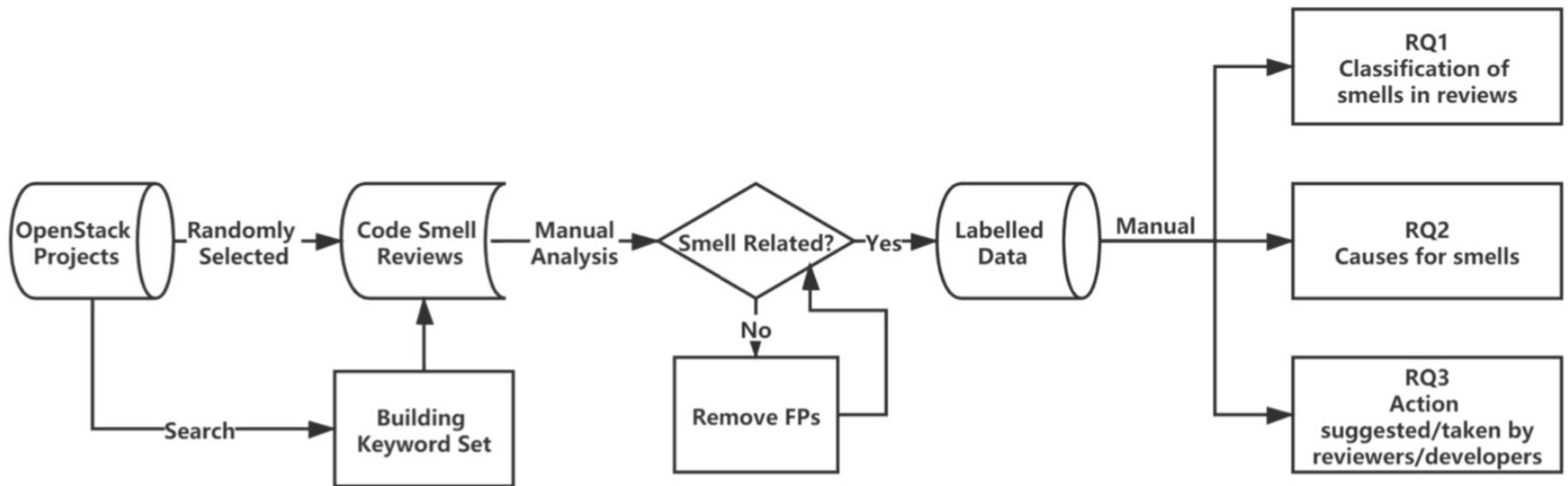


Fig. 1: An overview of our data mining and analyzing process

データセット

- プロジェクト : OpenStack (nova, neutron)
- プラットフォーム : Gerrit Code Review
- コードレビューデータはGerritのAPIから取得

TABLE I: An overview of the subject projects (Nova and Neutron)

Project	Review Period	#Code Changes	#Comments
Nova	Jan 14 - Dec 18	22,762	156,882
Neutron	Jan 14 - Dec 18	15,256	152,429
Total		38,018	309,311

キーワードセット作成

1. 初期設定したキーワードを単語に分ける
2. 単語を含むコメントを取得し，コメント内に含まれる単語を整形する
3. 文書行列作成
4. 初期キーワードと頻繁に共起する単語を見つけ，キーワードに追加

TABLE II: Code smell terms included in our mining

Code Smell Terms			
Accidental Complexity	Anti Singleton	Bad Naming	Blob Class
Circular Dependency	Coding by Exception	Complex Class	Complex Conditionals
Data Class	Data Clumps	Dead Code	Divergent Change
Duplicated Code	Error Hiding	Feature Envy	Functional Decomposition
God Class	God Method	Inappropriate Intimacy	Incomplete Library Class
ISP Violation	Large Class	Lazy Class	Long Method
Long Parameter List	Message Chain	Middle Man	Misplaced Class
Parallel Inheritance Hierarchies	Primitive Obsession	Refused Bequest	Shotgun Surgery
Similar Subclasses	Softcode	Spaghetti Code	Speculative Generality
Suboptimal Information Hiding	Swiss Army Knife	Temporary Field	Use Deprecated Components

一般的なキーワード
“code smell”, “bad smell”,
+ “bad pattern”, “anti-pattern”,
“technical debt”

レビューコメントの選定

1. 作成したキーワードセットで検索→18,082件
2. 目視で無関係と思われるレビューコメントを削除→3,666件
3. 目視でソースコードにレビューで指摘したコードスメルが確認できるものを選定→1,235件
4. 3で選定したものからコード変更URL, スメルタイプ, 議論をまとめられるものを選定→1,174件

ランダム選定

- キーワード検索からもれたレビューコメント
(291,229件) からランダムに選定 (1,064件)
- 先述の選定手順2～4を実施し, 16件のレビュー
コメントを選定
- 合計1,190件のコードスメル関連コードレビュー
コメントを分析に使用

各RQで実施する分析

- RQ1

- 選定手順4で記録した各スメルタイプの割合を計算

- RQ2

- MAXQDA*を関連するソースコードに使用してコードスメルの原因を出力
- 出力結果から主題分析

*<https://www.maxqda.com/>

各RQで実施する分析

- RQ3-1

- 目視で次の種類にカテゴライズ
 - Fix：リファクタリングなどの修正を提案
 - Capture：検出したが、修正の提案はない
 - Ignore：特定したスメルを無視するように提案

- RQ3-2

- プログラマが返答し、対応したソースコードに変更が加えられていることを確認

各RQで実施する分析

- RQ3-3
 - RQ3-1, RQ3-2の結果をもとに以下のようにカテゴライズ
 - レビューアーの提案にプログラマが同意
 - レビューアーの提案にプログラマが反対
 - レビューアーのコメントにプログラマが無反応

結果：RQ1（頻度）

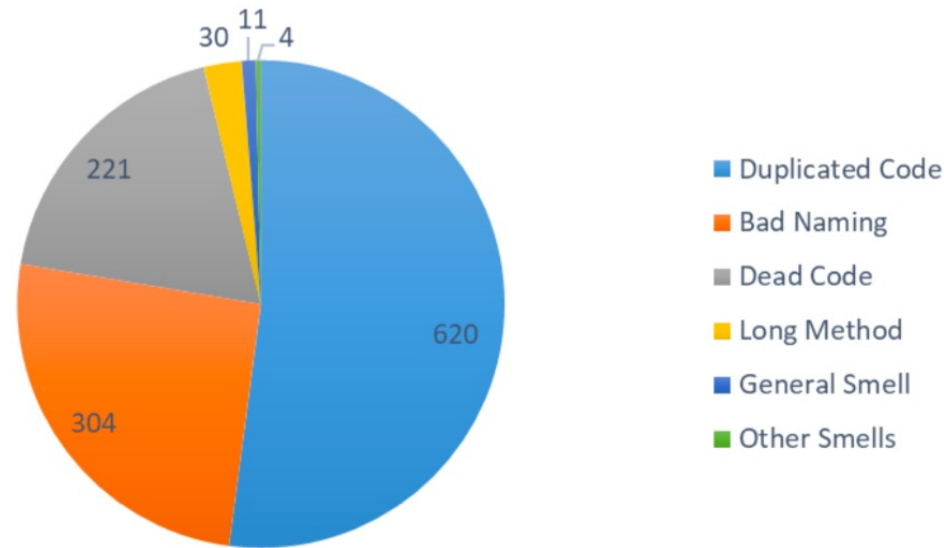


Fig. 3: Number of reviews for the identified code smells

- Duplicated Code, Bad Naming, Dead Code がほとんどであることがわかった

結果：RQ2（原因）

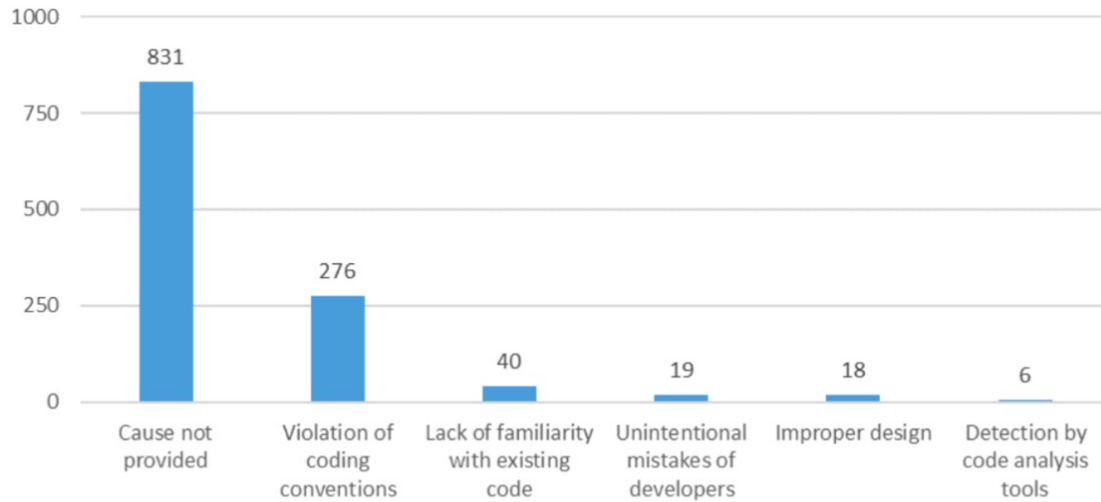


Fig. 4: Reasons for the identified smells

- 過半数が原因不明で、もっとも多かったのがコードデイング習慣違反

結果：RQ3（対処）

- ほとんどの提案はfixで、その35%は修正例を提示
- ほとんどのコードスメルは修正されている

TABLE III: Actions recommended by reviewers to resolve smells in the code

Reviewer's recommendation	Count
Fix (without recommending any specific implementation)	567
Fix (provided specific implementation)	303
Capture (just noted the smell)	272
Ignore (no side effects)	48

TABLE IV: Developers' actions to code smells identified during reviews

Code smell	#Reviews	#Fixed by developers	% of fixes
Duplicated Code	620	508	82%
Bad Naming	304	276	91%
Dead Code	221	210	95%
Long Method	30	25	83%
Circular Dependency	3	2	67%
Swiss Army Knife	1	1	100%
General Smell	11	7	64%
Total	1190	1029	86%

結果：RQ3（対処）

- ほとんどの場合レビュー通りにプログラマが対処している
- レビューへの反対は、修正の緊急性が低い、プロジェクトによる制限があるなどの理由が考えられる

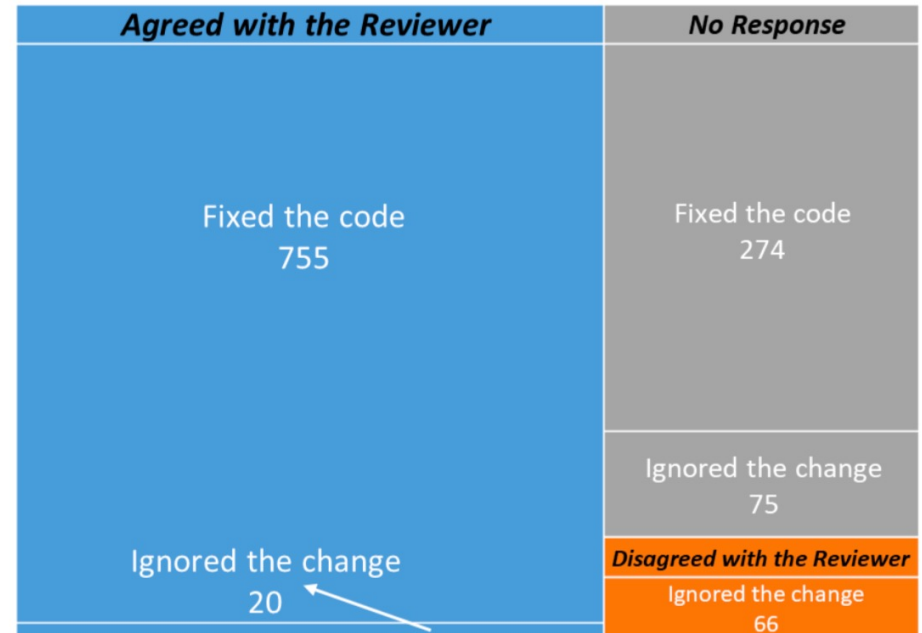


Fig. 6: A treemap of the relationship between developers' actions in response to reviewers' recommendations regarding code smells identified in the code

考察

- RQ1

- Duplicated Codeは既存研究でも多くの割合で存在することが報告されている[8,9]

- RQ2

- コーディング習慣違反が多いのはプログラマのドメイン知識不足が考えられる

- RQ3

- レビュー通りの修正が多いのは、レビュアーがドメイン知識が豊富で修正案として反映できるから

[8] A. Yamashita and L. Moonen, "Do developers care about code smells? an exploratory survey," in *Proceedings of the 20th Working Conference on Reverse Engineering (WCRE)*. IEEE, 2013, pp. 242–251.

[9] A.Tahir,J.Dietrich,S.Counsell,S.Licorish,andA.Yamashita,"Alarge scale study on how developers discuss code smells and anti-pattern in stack exchange sites," *Information and Software Technology*, vol. 125, 2020.

まとめ

- コードレビューによるコードスメル検出の有効性を調査
- コードレビューでは特定のスメルが高い割合で検出される
- レビューアーのドメイン知識が豊富であることで信頼性の高い修正案が提案される

所管

- コードレビューではDuplicated Codeが頻繁に議論されているのが驚きで、優先的に議論が必要な理由を知りたい
- コードレビュー段階で発生するDuplicated Codeの品質はどのようなものか気になる
- レビューアーが危険視しないパッチにコードスメルが含まれている場合、どのような種類か気になる