

Waiting around or job half-done? Sentiment in self-admitted technical debt

Gianmarco Fucci*, Nathan Cassee[†], Fiorella Zampetti*, Nicole Novielli[‡], Alexander Serebrenik[†], Massimiliano Di Penta*

*University of Sannio, Italy {gianmarco.fucci,fiorella.zampetti,dipenta}@unisannio.it

[†]Eindhoven University of Technology, The Netherlands {n.w.cassee,a.serebrenik}@tue.nl

[‡]University of Bari, Italy nicole.novielli@uniba.it

IEEE/ACM 18th International Conference on Mining Software Repositories (MSR), 2021,
pp. 403-414

SATD

- 実装が不完全であったり複雑であることを指し示すソースコードコメント
- 扱われるTDを理解するために、コメントが開発工程に則って分類されている^{[7][8]}

既存の分類における問題

- 他の工程との相互関係を考慮していない
 - 内容を加味した詳細な分類がされていない
- ➡ 問題解決の利用にはより細粒度な分類や分析が必要

[7] E. da S. Maldonado and E. Shihab, "Detecting and quantifying different types of self-admitted technical debt," in 7th IEEE International Workshop on Managing Technical Debt, MTD@ICSME 2015, Bremen, Germany, October 2, 2015, 2015, pp. 9–15.

[8] G. Bavota and B. Russo, "A large-scale empirical study on self-admitted technical debt," in MSR, 2016, pp. 315–326.

Research Question

- RQ1
 - SATDアノテーションはどんな問題を記述しているか
 - 従来のカテゴリよりも問題の細分化を図る
- RQ2
 - それぞれのSATDアノテーションの感情極性はどうか？
 - 対応の優先順位をつける際に、より否定的なコメントを残しているインスタンスの方が重要であると考え
- RQ3
 - 追加の情報をどの程度含んでいるか？
 - 従来研究のタスクアノテーション との比較

[14] M.-A. Storey, J. Ryall, R. I. Bull, D. Myers, and J. Singer, “Todo or to bug: Exploring how task annotations play a role in the work practices of software developers,” in Proceedings of the 30th International Conference on Software Engineering, ser. ICSE '08. New York, NY, USA: Association for Computing Machinery, 2008, p. 251–260.

データセット

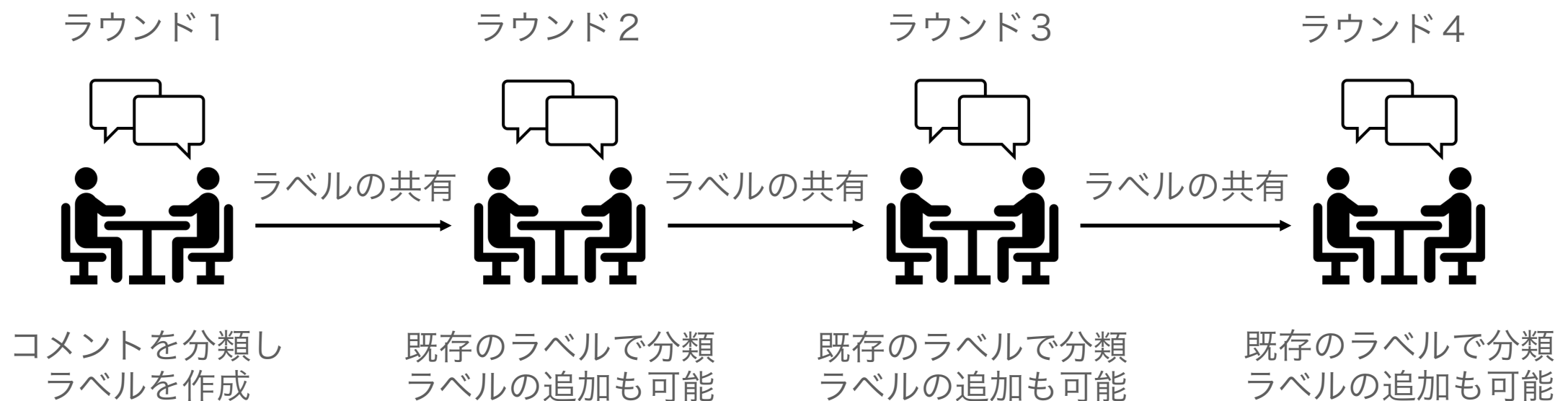
SATD Type	Initial Dataset	Without Duplication	Sampled
Defect	472	350	116
Design	2703	2260	657
Documentation	54	49	39
Implementation	757	550	183
Test	85	80	43
TOTAL	4071	3289	1038

- Maldonadoら^[2]によって公開されているデータセットを使用
 - 10のオープンソースJavaプロジェクトから収集
- 重複しているものを除き（782件）統計的サンプリングでサンプル抽出（最終的に1038件）
 - 信頼度 99%, 信頼区間 $\pm 3.33\%$

[2] E. da S. Maldonado, E. Shihab, and N. Tsantalis, "Using natural language processing to automatically detect self-admitted technical debt," *IEEE Trans. Software Eng.*, vol. 43, no. 11, pp. 1044–1062, 2017.

RQ1：アプローチ

- カードソーティングを利用
- 4ラウンドに分けて分類を実施



- 冗長なラベルを除去し， ラベルの親子関係を抽出

RQ1：結果

- 9の親カテゴリと32の子カテゴリに分類
 - 40件のコメントはFalse Positive
 - 俯瞰図は補足資料に記載

親カテゴリ	説明	件数
Poor implementation choice	最適でない実装	429
Partially/Not implemented functionality	部分的実装もしくは未実装	229
Functional issues	バグ	135
Wait	他の変更などの待ち	89
Documentation issues	ドキュメントの不足	54
Test issues	テストコード	36
Misalignment	要求と実装の齟齬	21
Outdated SATD comment	関係のないコメント	3
Deployment issues	デプロイ (jarファイル)	2
合計	—	998

従来の分類との比較

Macro-category	Defect	Design	Doc.	Impl.	Test	Total
Functional issues	48	68	0	18	1	135
Poor implementation choices	22	361	2	43	1	429
Wait	6	76	0	6	1	89
Deployment issues	1	0	0	1	0	2
SATD comments outdated	2	1	0	0	0	3
Partially implemented	27	94	5	100	3	229
Testing issues	0	1	0	0	35	36
Documentation issues	0	19	30	4	1	54
Misalignment	1	13	2	5	0	21
False positive	9	24	0	6	1	40
TOTAL	116	657	39	183	43	1038

- ほとんどのカテゴリが一对問題になっていない
- ➡ 各工程に当てはめても、その原因は一つではない
- テストやドキュメントは従来のカテゴリとほぼ一致

RQ2：アプローチ

- 手動でラベリング
 - Negative
 - ソースコードに対して否定的なコメント
 - Non-negative
 - ソースコードに対して肯定的あるいはどちらでもないコメント
 - Mixed
 - 肯定的なコメントと否定的なコメントの両方を含む
- 既存の感情分析ツールの利用を検討したが、否定的コメントの見逃しが多かったため不採択

RQ2 : 結果

Category	Negative	(%)	Non-negative	Mixed	Total
<i>Macro-categories in our taxonomy</i>					
Functional issues	66	(49%)	67	2	135
Poor implementation choices	125	(29%)	294	7	426
Wait	41	(46%)	45	3	89
Deployment issues	1	(50%)	1	0	2
SATD comments outdated	1	(33%)	2	0	3
Partially implemented	29	(13%)	197	2	228
Testing issues	12	(33%)	24	0	36
Documentation issues	18	(33%)	36	0	54
Misalignment	6	(29%)	15	0	21
<i>Categories of Maldonado et al. [2]</i>					
Defect	42	(40%)	60	4	106
Design	214	(34%)	407	9	630
Documentation	16	(41%)	23	0	39
Implementation	13	(7%)	163	1	177
Test	14	(33%)	28	0	42
TOTAL	299	(30%)	681	14	994

- Negativeなコメントが多いのはFunctional issues, Wait (ペアワイズ比較で他のカテゴリより有意に多いことを確認)
 - バグを含む場合や解決を待たされる場合に否定的

RQ3：アプローチ

- 次の情報がどのくらいの頻度で含まれるのかを確認
 - クラス名, メソッド名, プラグイン名, モジュール名
 - 開発者の名前やイニシャル
 - バグID
 - URL
 - 日時
- 全て手動で分析

RQ3 : 結果

Category	Component	Name	Bug id	URL	Date
Functional issues	47 (35%)	12 (9%)	11 (8%)	1 (1%)	9 (7%)
Poor implementation choices	152 (35%)	48 (11%)	5 (1%)	0	16 (4%)
Wait	21 (24%)	4 (5%)	9 (10%)	2 (2%)	1 (1%)
Deployment issues	0	0	0	0	0
SATD comments outdated	1 (33%)	0	1 (33%)	0	0
Partially implemented	50 (22%)	22 (10%)	0	0	1 (1%)
Testing issues	7 (19%)	3 (8%)	0	0	0
Documentation issues	19 (35%)	11 (20%)	0	0	1 (2%)
Misalignment	7 (33%)	4 (19%)	0	0	0
TOT. (UNIQUE)	304 (30%)	104 (10%)	26 (3%)	3 (0.3%)	28 (3%)

- 他の要素に比べてコンポーネント名は記述される傾向にある
 - トレーサビリティ向上のためであると考えられる
- URLや日時はITSやBTSを参照することで得られるため記述されにくい

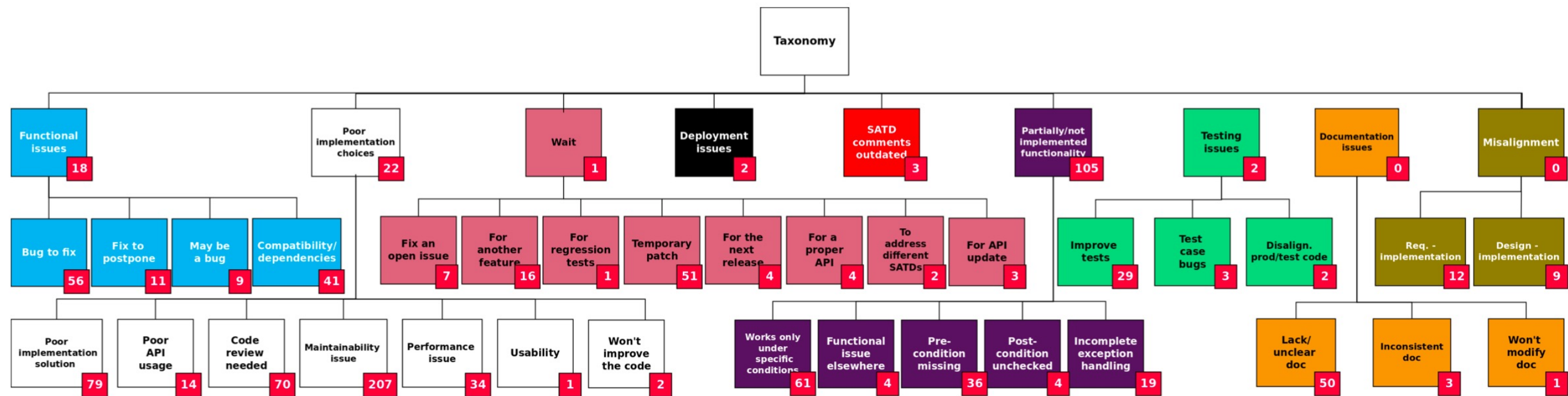
まとめ

- SATDコメントの詳細な分析を実施
 - 従来の分類（開発工程別）では複数の問題をまとめて取り扱っているカテゴリが存在する
 - バグや解決が待たされる場合に否定的なコメントが記述されやすい
 - 追加の詳細情報はあまり記述されないが、コンポーネント情報は記述されやすい傾向にある

所感

- 否定的なコメントは開発者にとって重要とされる可能性があるという考え方は思いつかなかった
- 今回は問題の細分化がメインだったが、問題と修正パターンの関係が調査されれば、よりTD解決に役立つ気がする
 - 修正パターンのイメージは全くない

付録：カテゴリの俯瞰図



- 色ごとにカテゴリ分け
- 数字は該当したコメントの件数