



Bug Propagation through Code Cloning: An Empirical Study

Author : Manishankar Mondal, Chanchal K. Roy, Kevin A. Schneider

Proceedings of the 33rd IEEE International Conference on Software Maintenance and Evolution 2017 Pages 227-237



目次

- 背景, 目的
- 拡散した欠陥の検出
- 拡散した欠陥の分析
- まとめ, 所感



背景, 目的

背景

- 開発者が気づかずに欠陥を含むコードを複製してしまうことがある
- 複製された欠陥の修正の手間が増える
- 欠陥がどのような割合でどのように拡散されるかはまだ明らかにされていない



目的

- どれほどの割合で欠陥が拡散するかを明らかにする
- 欠陥はどのような形で拡散するかを明らかにする



拡散した欠陥の検出

大まかな流れ

1. 対象プロジェクトのリビジョンからメソッドを取り出す
2. NiCad[5] でメソッドであるクローンを検出
3. クローン系譜を検出
4. 欠陥修正されたクローン片の検出
5. 拡散した欠陥の検出

クローン系譜を検出

- クローン系譜: VCS の中でクローンがどのように変化していくかを表したもの
- クローン系譜はタイプ別に分ける
- 途中でタイプが変わったクローンは異なる系譜として扱う

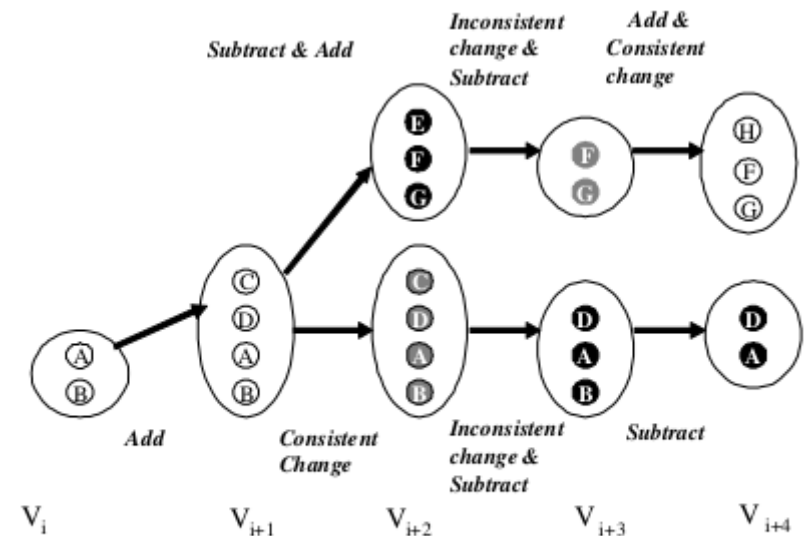


Figure 3: An example clone genealogy

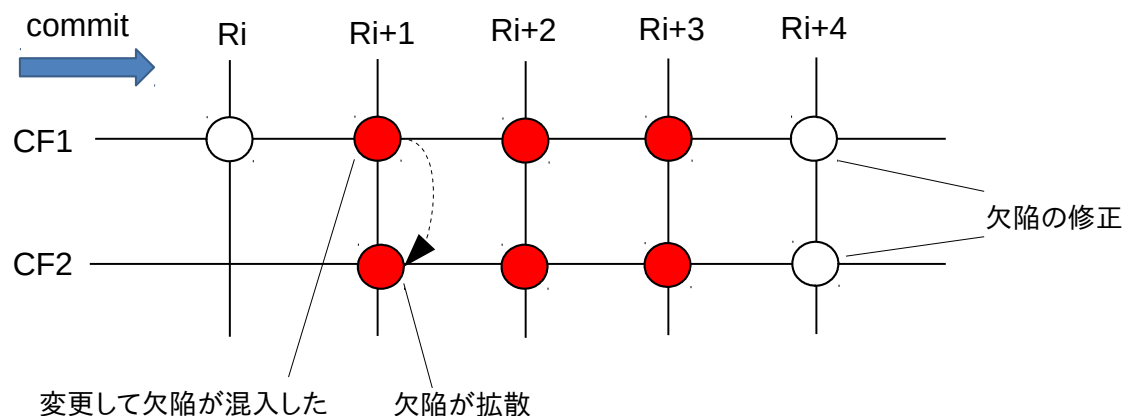
引用: An Empirical Study of Code Clone Genealogies Figure 3

欠陥修正されたクローン片の検出

- コミットメッセージの中に欠陥修正に関することが書いてあるものを見つけ出す
- そのリビジョンの修正箇所がクローン片に含まれていればそのクローンに欠陥が含まれていたことになる

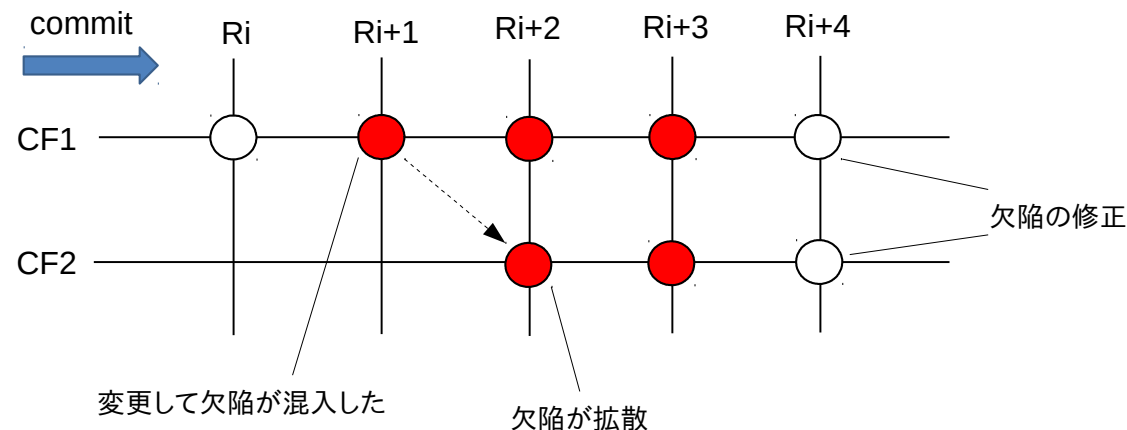
拡散パターン 1

- 同じリビジョンでクローンペアが生成された
- クローンペアが生成されてから修正されるまでに 1 回以下の変更があるもの
- 修正コミットで同時に修正された



拡散パターン 2

- あるリビジョンのコード片（クローン片）から後のリビジョンでコードが複製された
- 後で複製されたクローンが生成されてから修正されるまでに 1 回以下変更された
- 修正コミットで同時に修正された



拡散した欠陥の検出

- 4 の手順の中で同じリビジョンで欠陥修正されたクローンペアを選ぶ (パターン 1, 2 の 3 つ目の条件)
- そのクローンペアの系譜を見てどちらの拡散パターンの条件を満たすかどうかを調べる

拡散の例 (パターン 1)

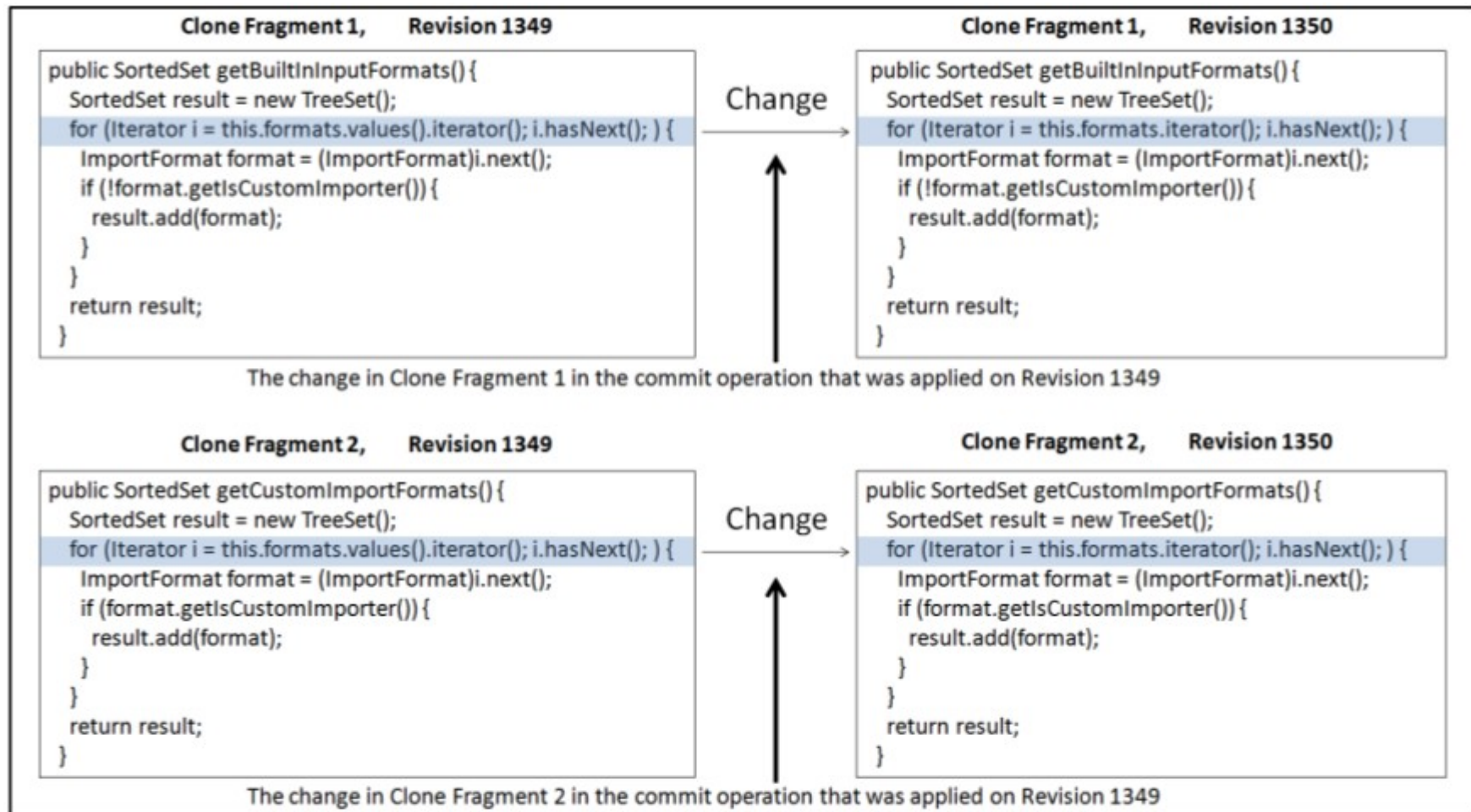


Fig. 2. The figure shows a similarity preserving co-change of two clone fragments (Clone Fragment 1, and Clone Fragment 2) in the commit operation which was applied on revision 1349 of our subject system Jabref. The commit operation that was applied on revision 1349 is a bug-fix commit. The commit message says 'JabRef 2.0: fixed some Bugs'. We provide the snapshots of the two clone fragments in two revisions, 1349 and 1350, and highlight the differences between the snapshots. From the figure it is clear that the two clone fragments contained the same bug, and the fragments were changed in the same way for fixing the bug. This bug is a propagated bug, because the clone fragments evolved by following the first bug-propagation pattern defined in Section IV. We see that the difference between the two clone fragments lies in the condition part of the *if-statement*. Clone Fragment 1 contains a NOT operator (!) which is absent in Clone Fragment 2.



拡散した欠陥の分析

データセット

TABLE II. SUBJECT SYSTEMS

Systems	Lang.	Domains	LLR	Revs
jEdit	Java	Text Editor	191,804	4000
Freecol	Java	Game	91,626	1950
Carol	Java	Game	25,091	1700
Jabref	Java	Reference Management	45,515	1545
LLR = LOC in the Last Revision			Revs = No. of Revisions	

リサーチクエストション

- RQ1 :それぞれのタイプのクローンで欠陥の拡散に関わりのあるものの割合はどれほどか
- RQ2 :それぞれのタイプのクローンに含まれる欠陥の内, 欠陥の拡散に関わりのあるものの割合はどれほどか
- RQ3 : 2 つの拡散パターンの内どちらが多いか

RQ1

- どれほどのクローンから欠陥が拡散するのか, 3 つのクローンタイプの内どのタイプで欠陥が拡散しやすいのかを明らかにするためにこの分析を行う
- 下の図にあるようにクローンタイプ別にクローン片や拡散した欠陥の総数を求めた

TABLE III. STATISTICS REGARDING BUG PROPAGATION IN DIFFERENT CLONE-TYPES

Systems	Type 1					Type 2					Type 3				
	CF	BC	BCF	BPCP	BPCF	CF	BC	BCF	BPCP	BPCF	CF	BC	BCF	BPCP	BPCF
Carol	415	8	31	1	2	211	8	32	2	4	682	22	134	130	39
Freecol	239	7	14	2	4	162	10	12	2	4	752	46	107	15	25
jEdit	7398	37	73	0	0	399	10	20	1	2	2688	42	184	4	8
Jabref	483	6	8	0	0	228	6	14	0	0	1363	23	31	1	2

CF = Total number of distinct clone fragments (i.e., clone genealogies) of a particular clone-type that were created during the whole period of evolution.

BC = Total number of bug-fix commits that affected clone fragments of a particular clone-type.

BCF = Total number of distinct clone fragments of a particular clone-type that experienced bug-fix commits (i.e., that experienced bug-fixes).

BPCP = Total number of distinct clone-pairs of a particular clone-type that evolved following a bug propagation pattern.

BPCF = Total number of distinct clone fragments of a particular clone-type that were evolved in the bug-propagation clone-pairs.

RQ1

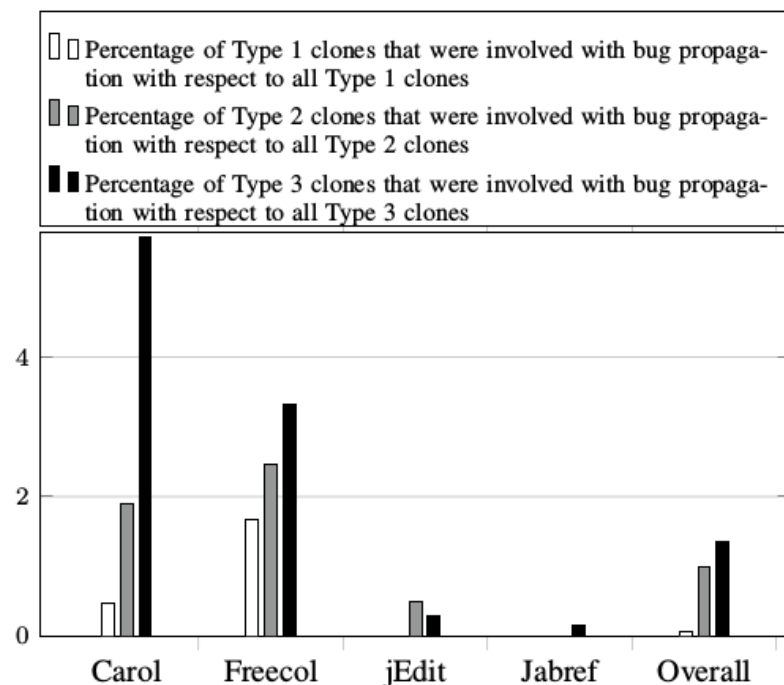


Fig. 3. Percentage of clone fragments that were involved with bug propagation with respect to all clone fragments considering each clone-type.

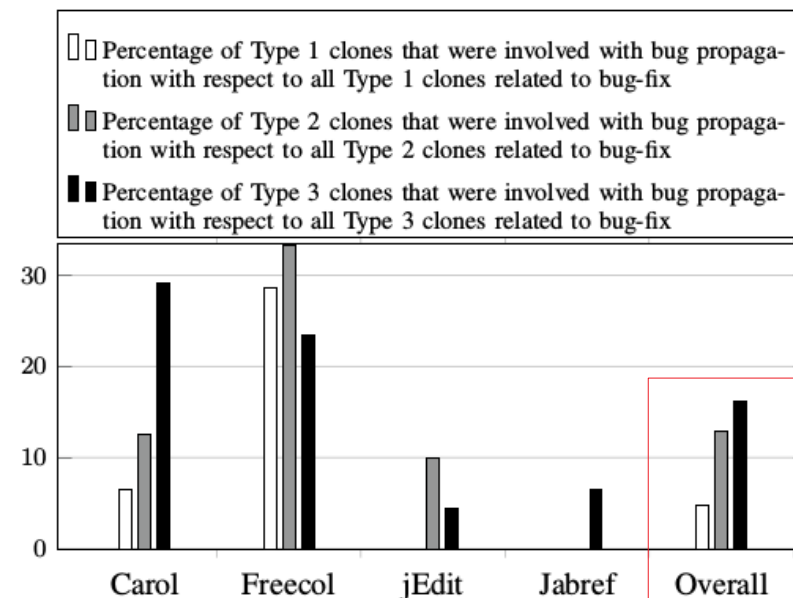


Fig. 4. Percentage of clone fragments that were involved with bug propagation with respect to all bug-fix clone fragments considering each clone-type.

全てのクローンタイプで33%
タイプ2, 3のクローンで拡散しやすいことが分かる

- 左の図は(拡散した欠陥 / クローン片)を求めたもの
- 右の図は(拡散した欠陥 / 欠陥混入クローン)を求めたもの

RQ2

- どれほどの欠陥がクローンによって拡散するかを明らかにするためにこの分析を行う
- 右の表は欠陥混入クローン修正コミットと拡散した欠陥を修正したコミットの総数を求めたもの
- 右の図は(拡散した欠陥を修正したコミット / 欠陥混入クローン修正コミット)を求めたもの

TABLE IV. STATISTICS REGARDING THE NUMBER OF PROPAGATED BUGS IN DIFFERENT CLONE-TYPES

Systems	Type 1		Type 2		Type 3	
	BC	BCPB	BC	BCPB	BC	BCPB
Carol	8	1	8	2	22	4
Freecol	7	2	10	2	46	9
jEdit	37	0	10	1	42	3
Jabref	6	0	6	0	23	1

BC = Total number of bug-fix commits that affected clone fragments of a particular clone-type.

BCPB = The number of bug-fix commits that indicate fixing of a propagated bug.

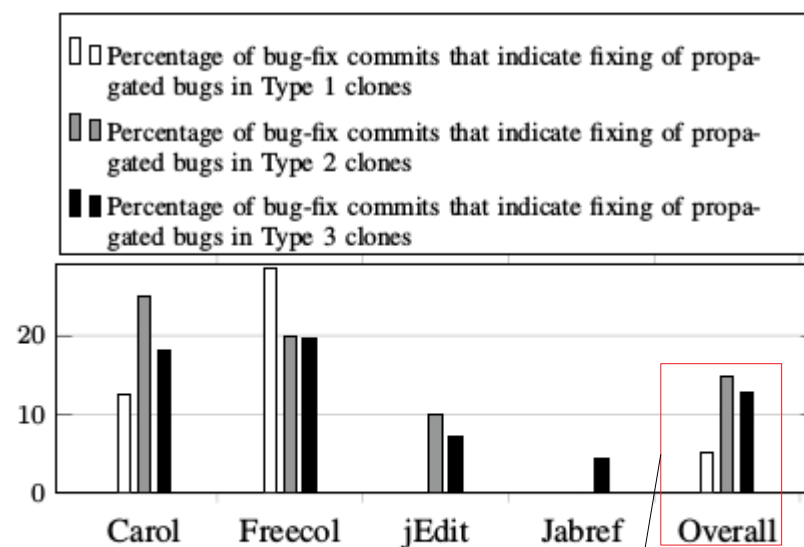


Fig. 5. Percentage of bug-fix commits that indicate fixing of propagated bugs in different clone-types

全てのクローンタイプ
で28.57% 19 / 23

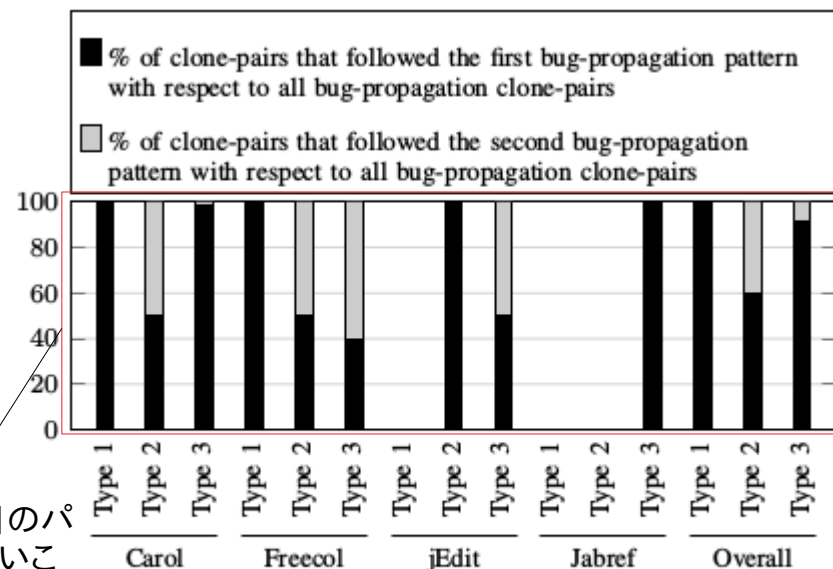
RQ3

- 2つの拡散パターンの内どちらに注意を向ければいいかを明らかにするためにこの分析を行う
- 右の表はそれぞれの拡散パターンの総数, 図の方はどちらが多いかを比較したもの

TABLE V. STATISTICS REGARDING THE NUMBER OF CLONE-PAIRS FOLLOWING DIFFERENT BUG PROPAGATION PATTERNS

Systems	Type 1		Type 2		Type 3	
	PF	PS	PF	PS	PF	PS
Carol	1	0	1	1	128	2
Freecol	2	0	1	1	6	9
jEdit	0	0	1	0	2	2
Jabref	0	0	0	0	1	0

PF = Number of clone pairs that followed the first pattern
PS = Number of clone pairs that followed the second pattern



全体的に1つ目のパターンの方が多いことが分かる

Fig. 6. Comparing the likeliness of occurrence of two bug propagation patterns.



まとめ, 所感

まとめ

- 欠陥を含むクローンの内, 欠陥が拡散したのは 33%
- 欠陥修正コミットの内, 拡散した欠陥と関係があるものは 28.57%
- 2 つの拡散パターンの内, 1 つ目のパターンの方が多い

所感

- 本研究では欠陥が同じリビジョンで修正されたものを扱っていたが、異なるリビジョンで修正されたものはどれくらいあるのか気になった
- 欠陥混入クローンの個数自体が小さいため、プロジェクトをもっと増やして同じ分析を行うとどうなるのか気になった