

CRDを用いたコードクローンの生存 期間と修正回数に関する調査

堀田 圭佑

肥後 芳樹

楠本 真二

コードクローンに関する研究

- コードクローンが持つ特性を理解
 - コードクローンを分析する研究が多数行われている
- 特に...
 - ソフトウェアの開発履歴情報の分析を通じて, コードクローンがどのような進化をたどっているのかを明らかにする手法

コードクローンの進化に関する既存研究

- Kimらの研究[10]
 - コードクローンの多くは生存期間が短い
 - コードクローンの除去による恩恵は小さいと主張
- Godeらの研究[11]
 - コードクローンのほとんどは1度も修正されないか, 1度しか修正されない
 - 複数回修正されるものは全体の約12%程度

[10] Kim, M., Sazawal, V., Notokin, D. and Murphy, G.C.: An Empirical Study of Code Clone Genealogies, Proc. 13th International Symposium on Foundations of Software Engineering, pp.187-196 (2005).

[11] Gode, N. and Koschke, R.: Frequency and Risks of Changes to Clones, Proc. 33rd International Conference on Software Engineering, pp.311-320 (2011).

対処がいるコードクローンの特性

- 生存期間が長いこと
 - 対処したコードクローンがすぐに消えると, かけた労力に見合う恩恵が得られないため
- 頻繁に修正されること
 - 修正されない安定したコードクローンへ対処をしても効果的でないため

既存研究で分かっていないこと

- 生存期間が長く、かつ多数修正されているコードクローンがどの程度存在するのか
- コードクローンがその生存期間のどの時点で修正されるのか
 - (例) 生成直後に頻繁に修正され、すぐに安定する場合
 - 生成されてから一定期間過ぎたものに対処しても効果的でない

既存研究による追跡手法の問題点

- コード片の類似度からコード片を対応付けて, コードクローンの追跡を行っていた
 - コード片に大きな修正があった場合
 - 対応付けができず, 追跡に失敗するおそれがある

本研究の目的

- コードクローンの追跡を行う際, 変更に強い追跡手法を用いることで, 既存研究で得られた知見に差異が生じるかを確認する
- 既存研究で明らかにされていないコードクローン進化の特性を明らかにする

RQ1～4

1. ‘コードクローンの多くは生存期間が短い’ という知見はCRDを用いた場合でも成立するか
2. ‘複数回修正されるコードクローンは少ない’ という知見はCRDを用いた場合でも成立するか
3. コードクローンの生存期間の長さとはコードクローンに加えられる修正回数に正の相関はあるか
4. コードクローンは生存期間の前半に修正されやすい傾向にあるか

コードクローンの検出手法

- ブロック単位のコードクローン検出手法を用いる
 - STEP1: 字句解析及び構文解析
 - STEP2: ブロックの特定
 - STEP3: 正規化
 - STEP4: ハッシュ値の算出
 - STEP5: コードクローンの特定

```
int method(int a, int b) {  
    int c = 0;  
    if (a == 0) {  
        c = c + 1;  
    }  
    if (b == 1) {  
        c = c + 2;  
    }  
    return a + b + c;  
}
```

STEP
1&2



```
int method(int a, int b) {  
    int c = 0;  
    IF (a == 0)  
    IF (b == 0)  
        return a + b + c;  
}
```

```
if (a == 0) {  
    c = c + 1;  
}
```

```
if (b == 0) {  
    c = c + 2;  
}
```

STEP
3&4&5



```
int method(int $, int $) {  
    int $ = $;  
    IF (a == 0)  
    IF (b == 0)  
        return $ + $ + $;  
}
```

```
if ($ == 0) {  
    $ = $ + $;  
}
```

```
if ($ == 0) {  
    $ = $ + $;  
}
```

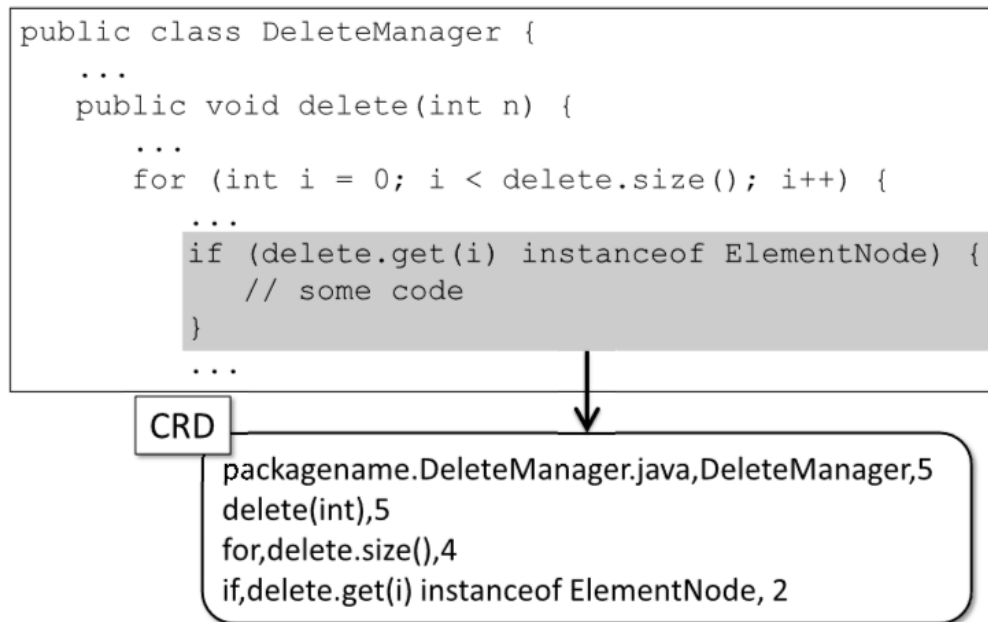
10

20

20

Clone Region Descriptor (CRD)

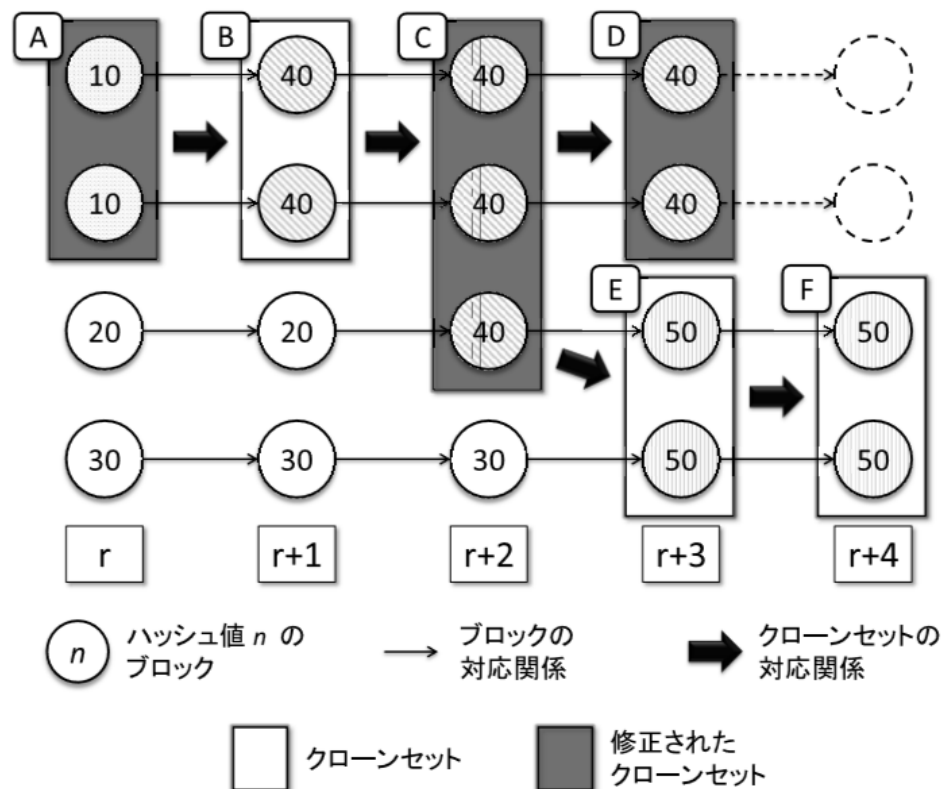
- Duala-Ekokoらが提案した, コードクローンを構成するコード片のおおよその位置情報を示す表現形式[17]



[17] Duala-Ekoko, E. and Robillard, M.P.: Clone Region Descriptors: Representing and Tracking Duplication in Source Code, ACM Trans. Software Engineering and Methodology, Vol.20, No.1, pp3:1-3:31 (2010).

コードクローンの追跡例

- コードクローンの系譜
 - 互いに対応関係にあるクローンセットの集合
- 例では, A, B, C, D, E, Fの6つのクローンセットで構成



調査手法の実装

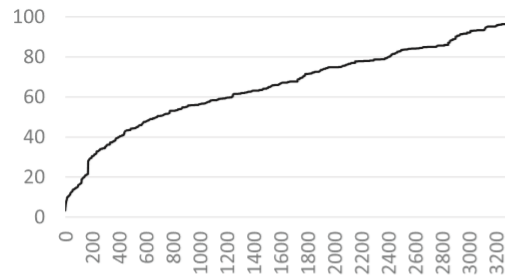
- 調査手法を実装ツールCloneTracker
- Javaで記述, かつバージョン管理システムSubversionを用いて管理されているソフトウェアのみ解析可能

調査対象

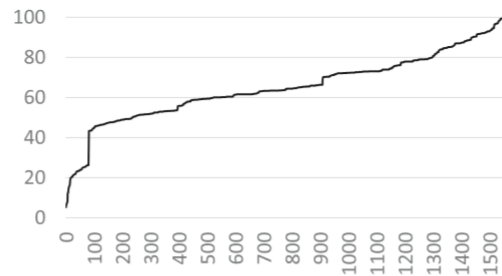
- Ant, ArgoUML: 開発期間が長く、また広く利用されているソフトウェア
- CloneTracker: 著者らが開発したソフトウェアで、詳細な分析ができる
- Carol, DNSJava: Kimらの研究[10]で用いられたソフトウェア
- JabRef: Godeらの研究[11]で用いられたソフトウェア

ソフトウェア	調査対象	開始リビジョン (日付)		終了リビジョン (日付)		リビジョン数	行数	系譜の数
Ant	/ant/core/trunk/src/main/	268,623	(2001/2/9)	909,962	(2010/2/14)	5,154	211,958	715
ArgoUML	/trunk/src/	1	(1998/1/27)	19,893	(2012/7/10)	3,918	362,783	2,760
CloneTracker	/	1	(2012/6/19)	419	(2013/5/3)	152	22,872	66
Carol	/trunk/carol/src/	9	(2002/8/6)	1,335	(2007/10/22)	250	17,251	258
DNSJava	/trunk/org/xbill/DNS/	2	(1998/9/6)	1,670	(2012/10/26)	1,285	22,512	406
JabRef	/trunk/jabref/src/java/net/	6	(2003/10/17)	3,718	(2011/11/11)	1,489	113,277	703

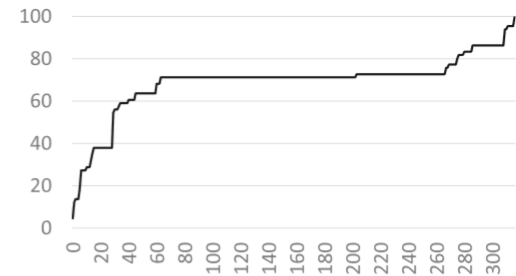
RQ1: ‘コードクローンの多くは生存期間が短い’ という知見はCRDを用いた場合でも成立するか



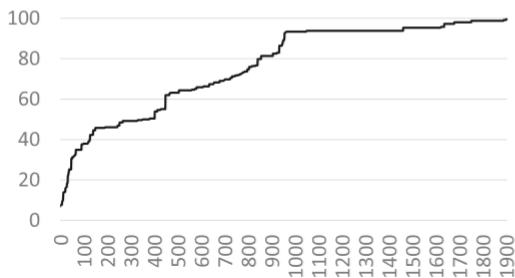
(a) Ant



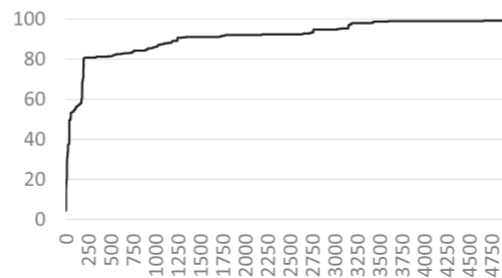
(b) ArgoUML



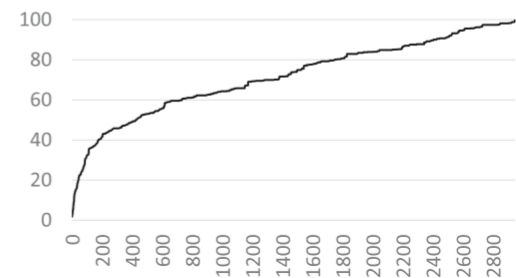
(c) CloneTracker



(d) Carol



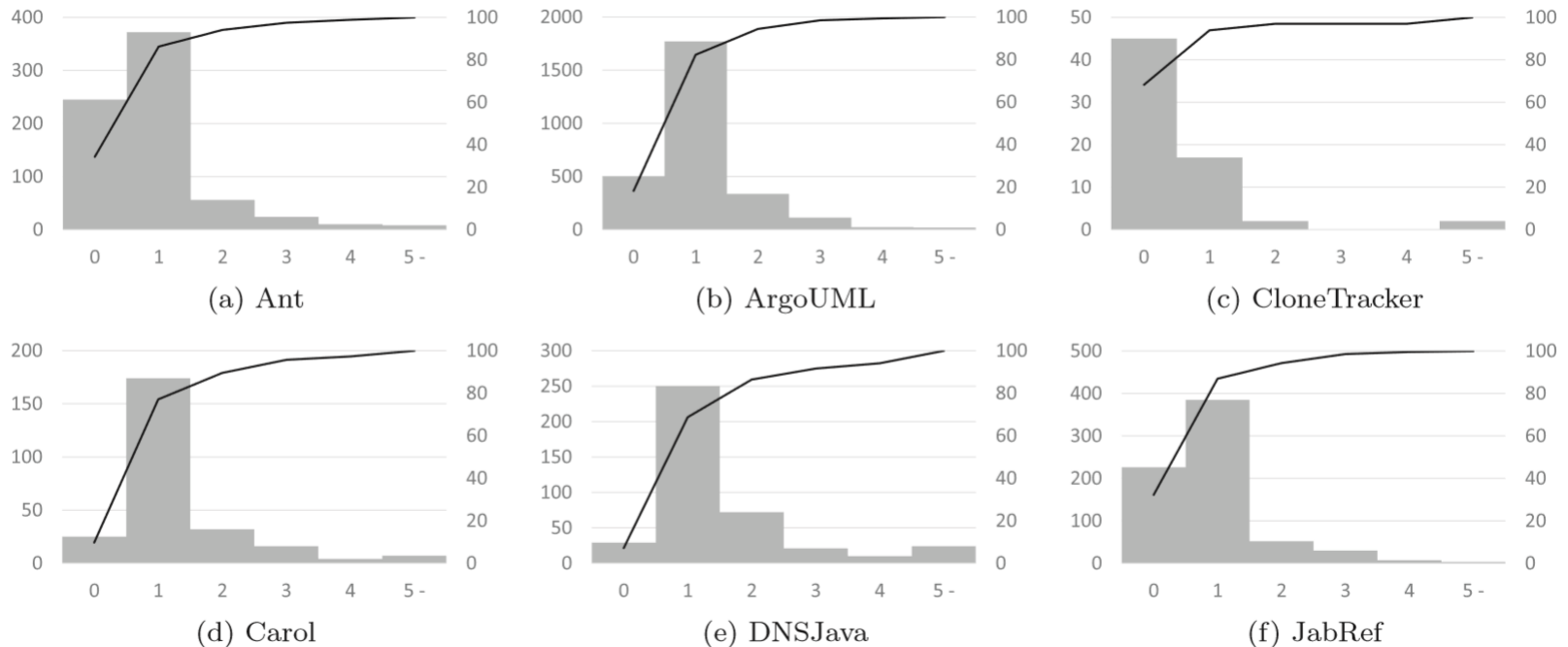
(e) DNSJava



(f) JabRef

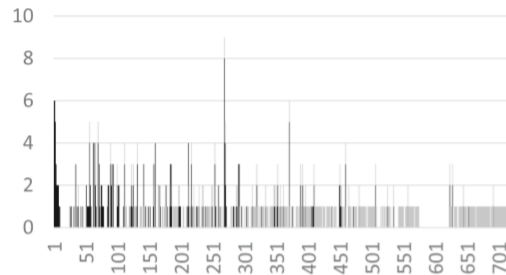
- x軸: 日数
- y軸: 生存日数がその日数以下である系譜数の百分率

RQ2: ‘複数回修正されるコードクローンは少ない’ という知見はCRDを用いた場合でも成立するか

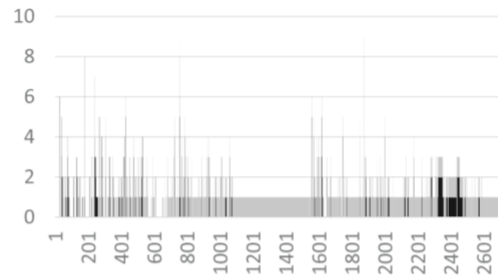


- x軸: 系譜に加えられた修正回数
- 棒グラフ(左軸): 修正回数がx軸の値である系譜の数
- 折れ線グラフ(右軸): 累積度数の百分率

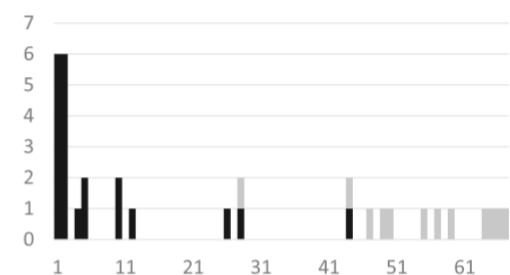
RQ3: コードクローンの生存期間の長さとコードクローンに加えられる修正回数の中に正の相関はあるか



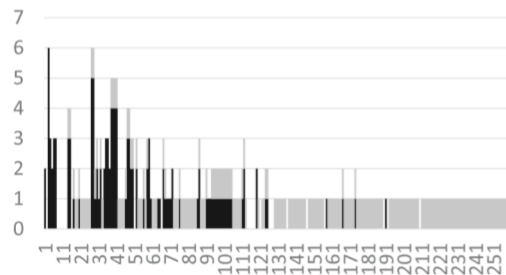
(a) Ant



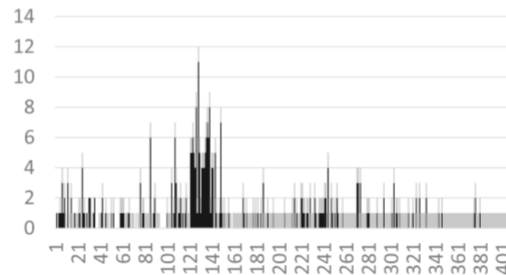
(b) ArgoUML



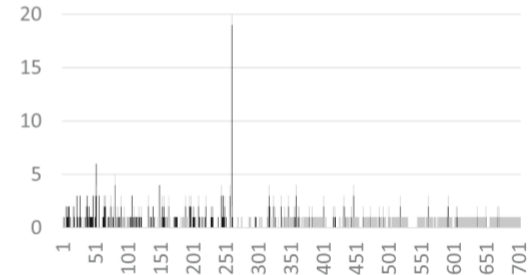
(c) CloneTracker



(d) Carol



(e) DNSJava



(f) JabRef

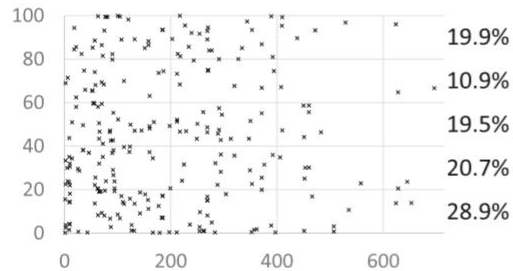
- x軸: 各系譜を生存期間(リビジョン数)の降順にならべたもの
- 棒グラフ: 系譜に加えられた修正の回数
 - グレーは消滅に関わる修正, 黒はそれ以外の修正の回数

RQ3: コードクローンの生存期間の長さとコードクローンに加えられる修正回数の中に正の相関はあるか

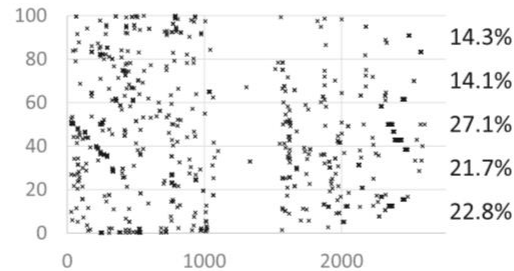
- 生存期間と修正回数の間に相関があるかをスピアマンの順位相関係数を用いて判定

	ソフトウェア	ρ	p -value
	Ant	-0.004515	0.9117
●	ArgoUML	-0.4130	2.2×10^{-16}
	CloneTracker	-0.05487	0.6617
●	Carol	0.3320	4.736×10^{-8}
	DNSJava	0.08566	0.08473
●	JabRef	-0.1393	0.0002124

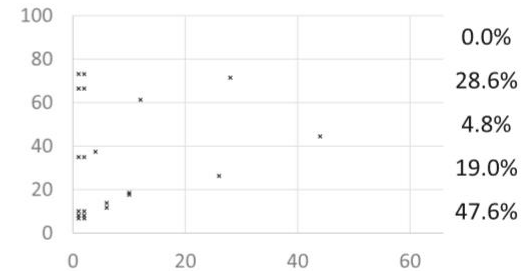
RQ4: コードクローンは生存期間の前半に修正されやすい傾向にあるか



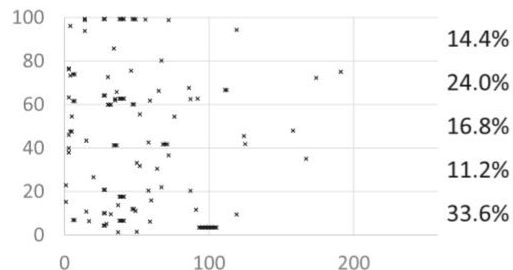
(a) Ant



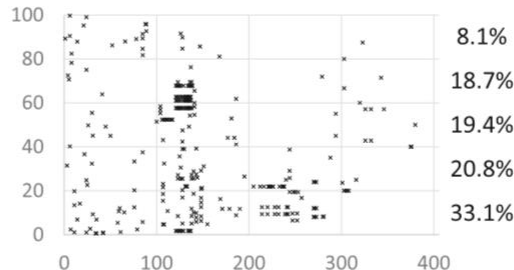
(b) ArgoUML



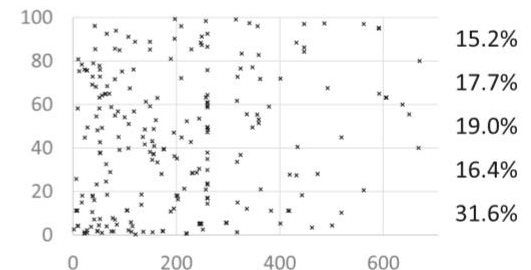
(c) CloneTracker



(d) Carol



(e) DNSJava



(f) JabRef

- x軸: 各系譜を生存期間(リビジョン数)の降順にならべたもの
- y軸: 修正を表す各点に対し, それぞれの修正が加えられた時期
- 右側の数値: それぞれのソフトウェアについて, その時期に加えられた修正の割合

RQ4: コードクローンは生存期間の前半に修正されやすい傾向にあるか

- 生存期間の前半と後半に加えられる修正回数に差があるか χ^2 検定を用いて判定

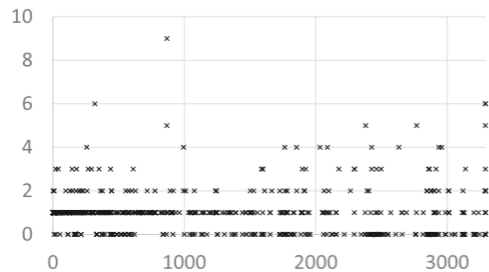
	ソフトウェア	前半	後半	χ^2	p -value
●	Ant	155	101	5.3407	0.02083
●	ArgoUML	540	305	32.7572	1.044×10^{-8}
	CloneTracker	15	6	1.2228	0.2688
	Carol	71	54	0.9042	0.3417
●	DNSJava	169	115	4.804	0.02839
	JabRef	133	98	2.3707	0.1236

考察1

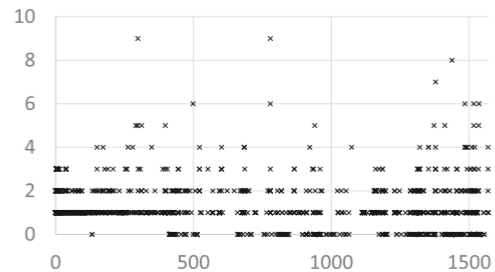
- 生存期間が長く, かつ修正回数の多いコードクローンがどの程度存在するか

ソフトウェア	生存期間	修正回数	生存期間&修正回数 (全系譜に占める割合)	
Ant	223	98	52	(7.27%)
ArgoUML	336	490	48	(1.74%)
CloneTracker	18	4	4	(6.06%)
Carol	54	59	30	(11.63%)
DNSJava	9	127	4	(0.99%)
JabRef	102	92	23	(3.27%)
計	742	870	161	(3.28%)

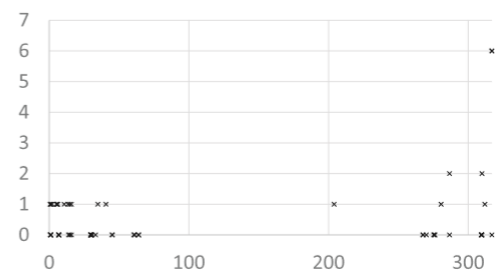
考察2



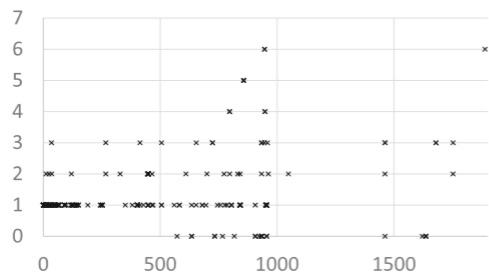
(a) Ant



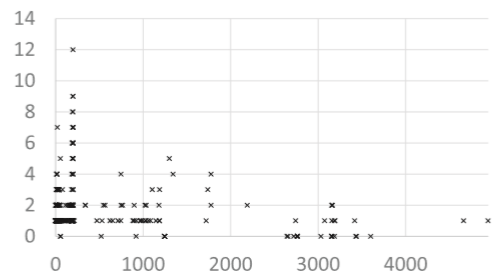
(b) ArgoUML



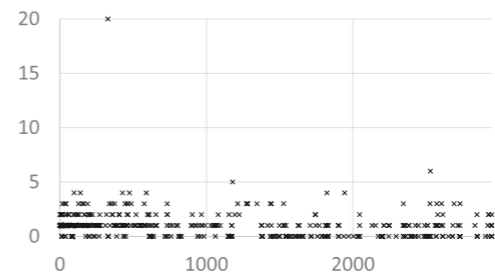
(c) CloneTracker



(d) Carol



(e) DNSJava



(f) JabRef

- x軸: 系譜の生存日数
- y軸: 修正回数

考察3

- 考察1, 2から, **生存期間が長く, かつ複数回修正されると**いう両方の特性を持つコードクローンは, いずれか一方の特性を持つコードクローンと比較して少ない
- すべてのコードクローンに対し平等に対処を行うことは効率的ではなく, **両方の特性を持つ**対処すべきコードクローンを慎重に選択する必要がある

主観

- そういう考え方があるという参考になった
 - 生存期間の前半など, 期間のどの時点かを考えていること