

An Empirical Study of Code Clone Genealogies

著者 Miryung Kim, Vibha Sazawal
David Notkin, Gail C. Murphy

発表者 野口耕二郎

ESEC/FSE' 05: Proceedings of the 10th European software engineering conference held jointly with 13th ACM SIGSOFT international symposium on Foundations of software engineering, pages 187-196

目次

- 背景, 目的
- クローン系譜
- 分析結果
- まとめ, 所感

背景, 目的

背景

- コードの重複は「コードの不吉な臭い」として、リファクタリングを行う対象とされている
- コードクローンを検出、除去する研究が盛んに行われてきた
- しかし、その当時は本当にすべてのコードクローンを除去することに価値があるのかについて研究はされていなかった

目的

- コードクローンの更なる分析のために, クローン系譜を定義し自動的に検出できるようにする
- クローン系譜を分析し, どのようなコードクローンに対してリファクタリングを行うべきかを明らかにする

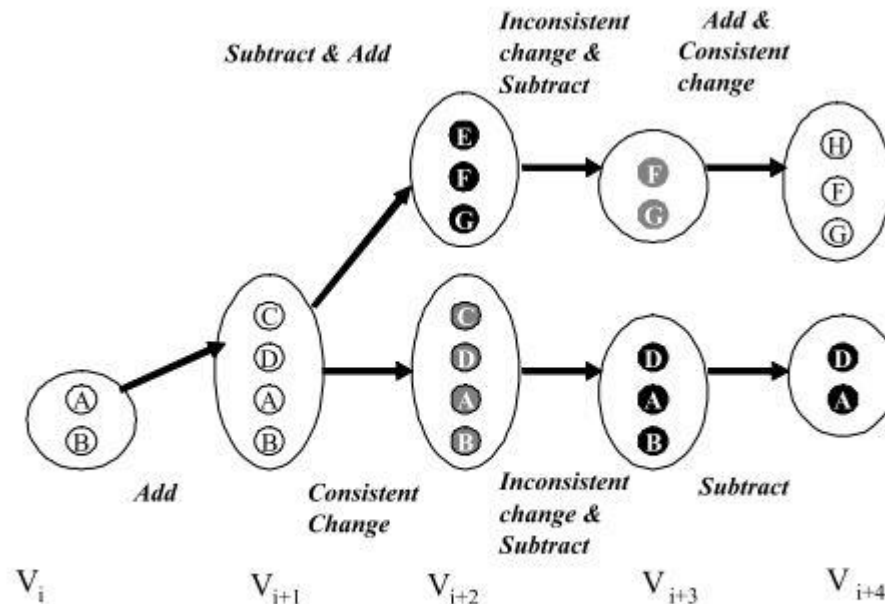
クローン系譜

コードクローン

- ソースコードの中的一致または類似したコード片のペアまたは集合
- クローングループ: 同じコードクローンのコード片の集合
- クローングループの中に1つでもバグが混入していると, クローングループすべてのコード片を修正する必要がある

クローン系譜の定義

- バージョン数が増えていく中で、クローングループ内部のそれぞれのコード片がどのように変化したかを表したクローングループの歴史



Add: クローングループに新たにいくつかのコード片が加わった
Subtract: クローングループからいくつかのコード片が消えた
Consistent Change: クローングループ内のコード片がすべて同じように変更された
Inconsistent Change: クローングループの内すべてではなく少なくとも1つが変更された

Figure 3: An example clone genealogy

クローン系譜の検出

- 以下の3つを入力として扱う
 - 年代順に並べられた複数のバージョン
 - クローンの検出結果
 - コード片の位置情報
- クローンの検出で用いたツールと同じものを用いてバージョン間のクローングループ内部のそれぞれのコード片の関係を調べる
- バージョン間のクローングループの内部のコード片の関係を調べた後、新しいほうのバージョンのクローングループを系譜につなげる

分析結果

分析項目

- 一貫性を持って変更されたクローン系譜
- 短命なクローン系譜
- 標準的なリファクタリングが使えないクローン
- 長命なクローン系譜

データセット

- 対象となるプロジェクト
 - carol, dnsjava
- 期間
 - carol: 2002年8月～2004年10月の26ヶ月
 - dnsjava: 2003年4月～2004年11月の19ヶ月
- バージョン数(クローンに対して変更が当たられたものだけ)
 - carol: 37/164バージョン
 - dnsjava: 224/905バージョン
- 検出されたクローン系譜
 - carol: 109個
 - dnsjava: 49個

一貫性を持って変更された クローン系譜

- 一貫性を持って変更されたということは、バグが混入するごとにグループ内すべてのクローンを変更する必要があったということ
- 結果
 - carol: 38%
 - dnsjava: 27%
- 比較的多くのクローンが一貫性を持って変更されてきたことがわかる

短命なクローン系譜

- 短命なクローン系譜を見るために途絶えたクローン系譜の個数を調べる
- 途絶えたクローン系譜の個数
 - carol: 53/109個
 - dnsjava: 31/49個
- k-volatile genealogies
 - バージョンkまでに途絶えたクローン系譜の数
 - $k\text{-volatile genealogies} = \{g \mid g \text{ is a dead genealogy and } 0 \leq g.\text{age} \leq k\}$

短命なクローン系譜

- 途絶えたクローン系譜をより分析するために様々な関数を定義
 - $f(k)$: バージョン k で存在するクローン系譜の個数
 - $f_{dead}(k)$: バージョン k で途絶えたクローン系譜の個数
 - $CDF_{dead}(k) = \frac{\sum_{i=0}^k f_{dead}(i)}{\sum_{i=0}^n f_{dead}(i)}$
 - 全体の途絶えたクローン系譜の内 k -volatile genealogies の割合
 - $R_{volatile}(k) = \frac{\sum_{i=0}^k f_{dead}(i)}{\sum_{i=0}^n f(i)}$
 - 全体のクローン系譜の内 k -volatile genealogies の割合

短命なクローン系譜

- 左図は k -volatile genealogiesの平均的な寿命を示したもの
- 右図は $CDF_{dead}(k)$ と $R_{volatile}(k)$ の計算結果を示したもの

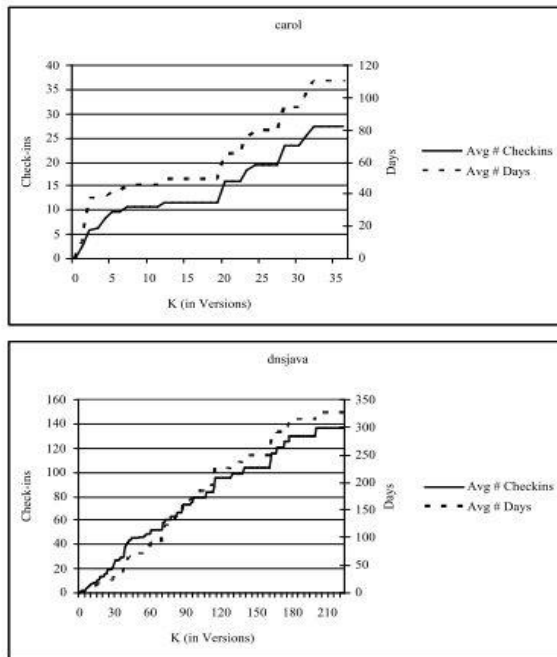


Figure 4: The average lifetime of k -volatile clone genealogies

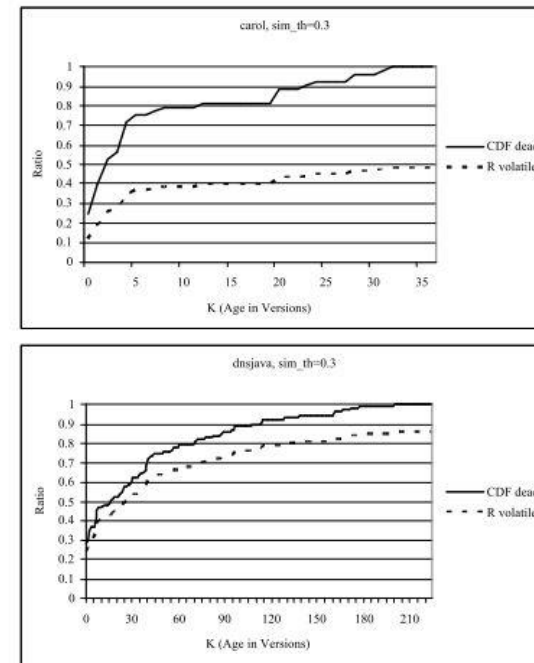


Figure 5: $CDF_{dead}(k)$ and $R_{volatile}(k)$ of *carol* and *dnsjava*

短命なクローン系譜

- 図5から, バージョン5までには全体の系譜の内
carolで37%, dnsjavaで31%
のクローン系譜が途絶えていることがわかる
- さらに図4から, バージョン5までに
carolで37%のクローン系譜が平均9.6チェックイン
dnsjavaで31%のクローン系譜が平均1.48チェックイン
で途絶えることがわかる
- 以上から比較的多くのクローン系譜が短命であることがわかり, それらのクローンにリファクタリングを行うのはあまり価値がないことがわかる

標準的なリファクタリングが 使えないクローン

- 以下の何れかに該当すれば標準的なリファクタリング使用不可能
 - 標準的なリファクタリングを行ってもクローンを除去できない
 - 組み合わせたリファクタリングを用いないとクローンを除去できない
 - 言語の制約によりクローンを除去できない
- このようなクローン系譜の割合は
 - carol: 64%
 - dnsjava: 49%

Table 3: Code Example of Locally Unfactorable Clones

<pre>public void exportObject(Remote obj) throws RemoteException{ if (TraceCarol.isDebugEnabledRmiCarol()) { TraceCarol.debugRmiCarol("MultiPRODelegate.exportObject(") } try { if (init) { for (Enumeration e = activePtcls.elements(); ... ((ObjDlgt)e.nextElement()).exportObject(obj); } } else { initProtocols(); //iterate protocol elements and export obj } } catch (Exception e) { String msg = "exportObject(Remote obj) fail"; TraceCarol.error(msg,e); throw new RemoteException(msg); } }</pre>	<pre>public void unexportObject(Remote obj) throws NoSuchObjectException { if (TraceCarol.isDebugEnabledRmiCarol()) { TraceCarol.debugRmiCarol("MultiPRODelegate.unexportObject(") } try { if (init) { for (Enumeration e = activePtcls.elements(); ... ((ObjDlgt)e.nextElement()).unexportObject(obj); } } else { initProtocols(); //iterate protocol elements and unexport obj } } catch (Exception e) { String msg = "unexportObject(Remote obj) fail"; TraceCarol.error(msg,e); throw new NoSuchObjectException(msg); } }</pre>
--	---

長命なクローン系譜

- リファクタリングの対象となる長命なクローン系譜の中にどれだけリファクタリング不可能なクローンが含まれているかを調べる必要がある
- 分析期間の半分以上存続した系譜とその中でリファクタリング不可である系譜の個数
 - carol: 長命な系譜が37個, その中でリファクタリング不可であるのは24個
 - dnsjava: 長命な系譜が18個, その中でリファクタリング不可であるのは13個
- 長命な系譜の内, リファクタリング不可かつ, 一貫性を持って変更されてきた系譜の個数
 - carol: 19個
 - dnsjava: 11個

長命なクローン系譜

- 右図はリファクタリング不可なクローン系譜と一貫性を持って変更されたクローン系譜とその両方の性質を持つクローン系譜の累積分布を示したもの

- リファクタリング不可であるクローン系譜が長命なクローン系譜の中で多くの割合を示しており、除去できるクローンが限られていることが分かる

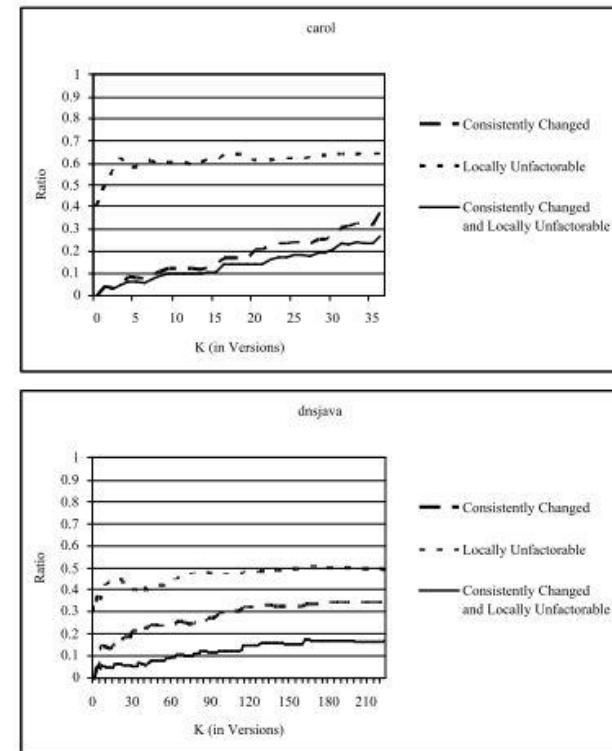


Figure 6: Cumulative fraction of consistently changed genealogies, locally unfactorable genealogies, and consistently changed and locally unfactorable genealogies

まとめ, 所感

まとめ

- リファクタリングを行う価値がないような短命なクローン系譜が比較的多く存在していることがわかった
- 長命なクローン系譜の中でリファクタリング不可であるものが多くを占め、除去可能なクローンは限られてくることがわかった
- 以上から、コードクローンに対して無闇にリファクタリングを行うことで不利益を被ることがわかる

所感

- 無闇にコードクローンを除去するのはかえって不利益を被ることになるということは、開発者にとってかなり有益な情報であると思った
- コードクローンのより詳しい性質を知ることができ、コードクローンをどのように扱えばよいかをより深く考えることができるようになった
- リファクタリング不可であるクローンについてもより詳しく知りたいと思った