Lecture 26
Empirical Studies of Clone Evolution
Clone Genealogies
Today’s Agenda (1)

- Class Presentation
  - Meiru Che
  - Amal Banerjee
- Course Evaluation
  - I need a volunteer to collect and deposit course evaluation forms.
Today’s Agenda (2)

- Discussion on practical implications of SE research
- Discussion on “An Empirical Study of Clone Genealogies”
Recap of CCFinder

• CCFinder is a robust and scalable clone detector.
• It transforms a program to a parameterized token sequence using language dependent transformation rules.
• It then use a suffix tree algorithm to find common contiguous subsequences.
• Its case studies show that CCFinder can be applied to industrial size programs.
Class Presentations

- Advocate: Meiru
- Skeptic: Amal
Course-Instructor Survey

- Instructor’s Name: Kim, Miryung
- This survey is for the instructor, not TA.
- Course Abbreviation and Number: EE382V Software Evolution
- Course Unique Number: 16730
- Semester and Year: Spring 2009
Discussion - Refactoring

• What is a definition of refactoring?
Discussion - Information Hiding

- What did you learn from the class activity on refactoring?
- (1) What do you need to consider before restructuring a program?
Discussion - Information Hiding

• What did you learn from the class activity on refactoring?

• (2) What do you need to consider after restructuring a program?
Discussion - Information Hiding

- What is the Information Hiding Principle?
Discussion - Information Hiding

• How can you apply the Information Hiding Principle to your software design process?

•
Program Differencing

- Which tool do you currently use to compare program versions?

- Why is program differencing important in software evolution research?
Program Differencing

• In this course, you have studied many different types of program differencing tools, such as diff, AST-based diff, Jdiff, UMLDiff, and LogicalStructuralDiff.

• (1) Pick one of the above tools and describe its key ideas and benefits of using it.
Program Differencing

• In this course, you have studied many different types of program differencing tools, such as diff, AST-based diff, Jdiff, UMLDiff, and LogicalStructuralDiff.

• (2) How will you apply these key ideas in the absence of the program differencing tool that can run on your codebase?
Clone Genealogy

- An Empirical Study of Code Clone Genealogies, Kim et al. ESEC/FSE 2005
- Studies of code clone evolution
- Mining software repositories research
- Its study results challenged one of the most widely-held conventional wisdom about clones.
Conventional Wisdom

**Code clones indicate bad smells of poor design. We must aggressively refactor clones.**

```java
public void updateFrom (Class c) {
    String cType = Util.makeType(c.Name());
    if (seenClasses.contains(cType)) {
        return;
    }
    seenClasses.add(cType);
    if (hierarchy != null) {
        ....
    }
    ...
}
```

```java
public void updateFrom (ClassReader cr) {
    String cType = CTD.convertType(c.Name());
    if (seenClasses.contains(cType)) {
        return;
    }
    seenClasses.add(cType);
    if (hierarchy != null) {
        ....
    }
    ...
}
```
Our Previous Study of Copy and Paste Programming Practices at IBM

• Even skilled programmers often create and manage code clones with clear intent.
  - Programmers cannot refactor clones because of programming language limitations.
  - Programmers keep and maintain clones until they realize how to abstract the common part of clones.
  - Programmers often apply similar changes to clones.

[Kim et al. ISESE2004]
Research Questions

How do clones evolve over time?

• consistently changed?
• long-lived (or short-lived)?
• easily refactorable?
Previous Studies of Code Clones

- automatic clone detection
  - lexical, syntactic (AST or PDG), metric, etc.

- studies of clone coverage ratio
  - gcc (8.7%), JDK (29%), Linux (22.7%), etc.

- studies of clone coverage change
  - changes of clone coverage in Linux [Antoniol+02], [Li+04]

These studies do not answer how individual clones changed with respect to other clones.
Outline

- motivation
- clone genealogy: model and tool
- study procedure and results
Model of Clone Evolution

Version $i$

Version $i+1$

Version $i+2$

Version $i+3$

Clone group

Code snippet

Location overlapping relationship

Cloning relationship

Add
Consistent Change
Inconsistent Change

Evolution Patterns
Clone genealogy is a set of clone groups connected by cloning relationships over time.

consistently changed

copied, pasted, and modified

lineage

lineage
Clone Genealogy Extractor (CGE)

Given multiple versions of a program, $V_k$ for $1 \leq k \leq n$.

- find clone groups in each version using CCFinder.
- find cloning relationships among clone groups of $V_i$ and $V_{i+1}$ using CCFinder.
- map clones of $V_i$ and $V_{i+1}$ using diff based algorithm.
- separate each connected component of cloning relationships (a clone genealogy).
- identify clone evolution patterns in each genealogy.
Outline

- motivation
- clone genealogy : model and tool
- study procedure and results
## Two Java Subject Programs

<table>
<thead>
<tr>
<th>Program</th>
<th>carol</th>
<th>dnsjava</th>
</tr>
</thead>
<tbody>
<tr>
<td>LOC</td>
<td>7878 ~ 23731</td>
<td>5756 ~ 21188</td>
</tr>
<tr>
<td>Duration</td>
<td>2 years 2 months</td>
<td>5 years 8 months</td>
</tr>
<tr>
<td>versions</td>
<td>37</td>
<td>224</td>
</tr>
</tbody>
</table>

versions: a set of check-in snapshots that increased or decreased the total lines of code clones