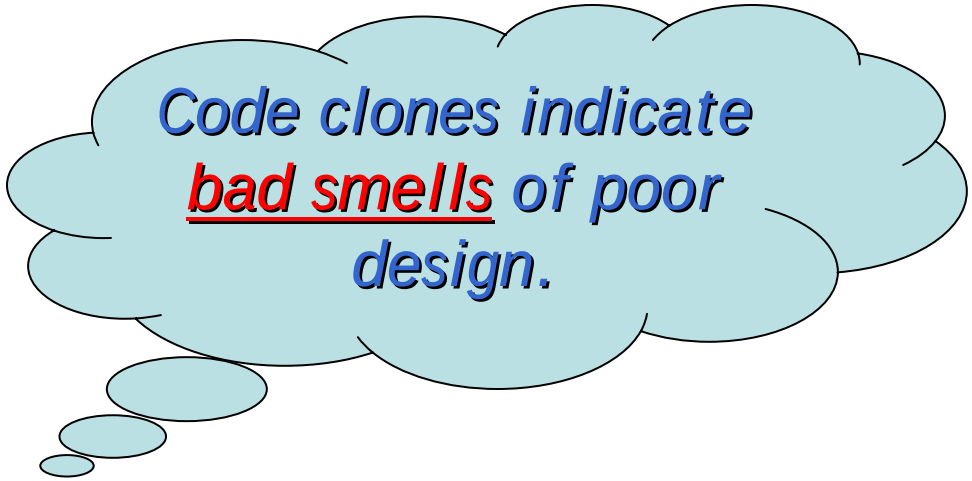
Using a *Clone Genealogy Extractor* for Understanding and Supporting Evolution of Code Clones

Miryung Kim and David Notkin

May 17th, 2005

University of Washington
Computer Science & Engineering

Conventional Wisdom




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

```
public void updateFrom  (ClassReader cr )  {
      String cType =CTD.convertType (c.Name());
      if (seenClasses.contains(cType)) {
         return;
      }
      seenClasses.add(cType);
      if (hierarchy != null) {
        ....
      }
      ...
```

A Study of Copy and Paste Programming [Kim et al. 2004]

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

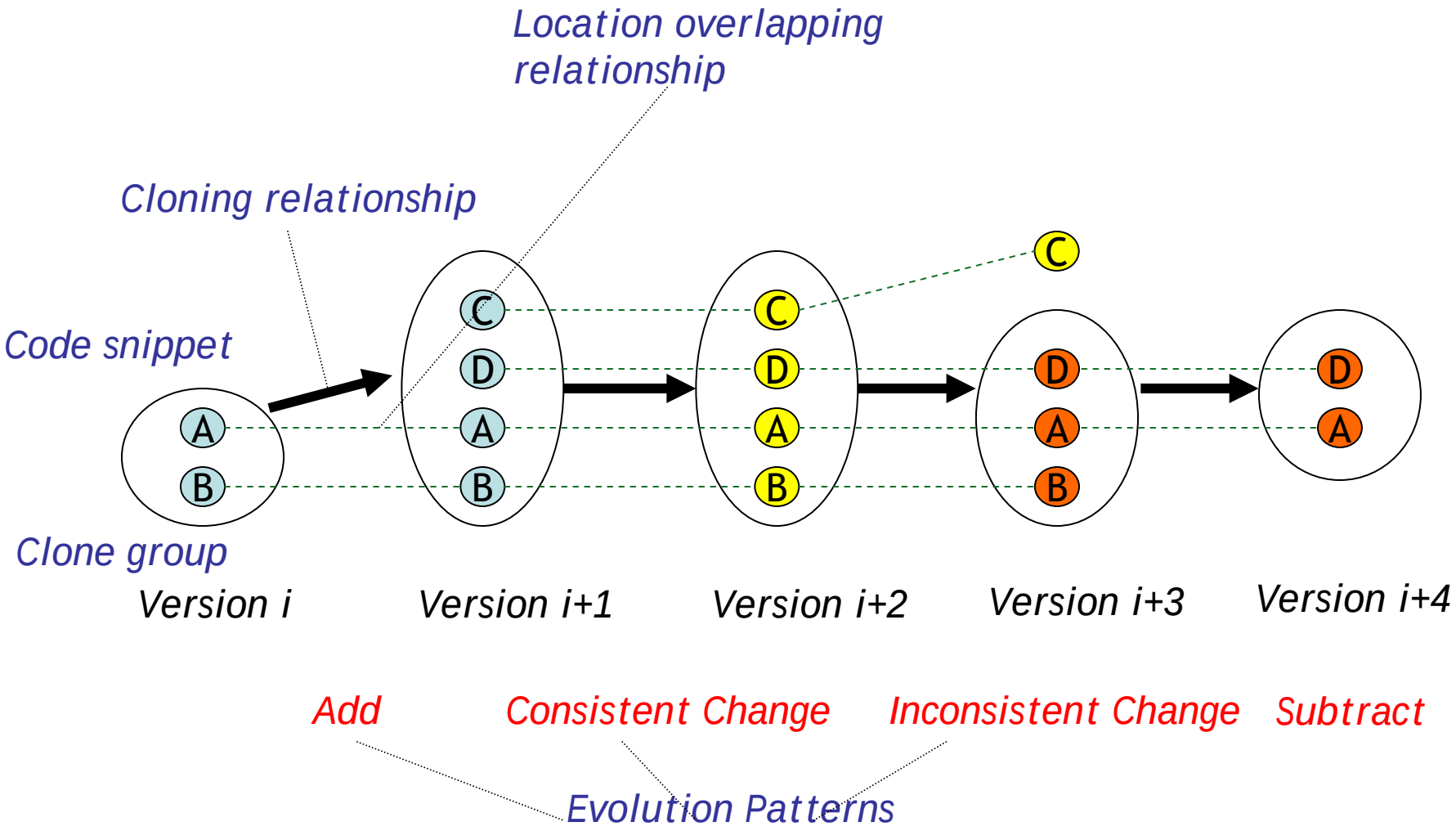
Gathering Data from Other Existing Software Projects

	What we captured for our copy and paste study	What most existing software projects provide in reality
Data source	editing logs captured by an Eclipse plug-in	version control system or release archives
Scale	a single programmer workspace	multi programmer collaborative environment
Granularity	key strokes and editing operations such as copy, cut, and paste	check-ins or releases

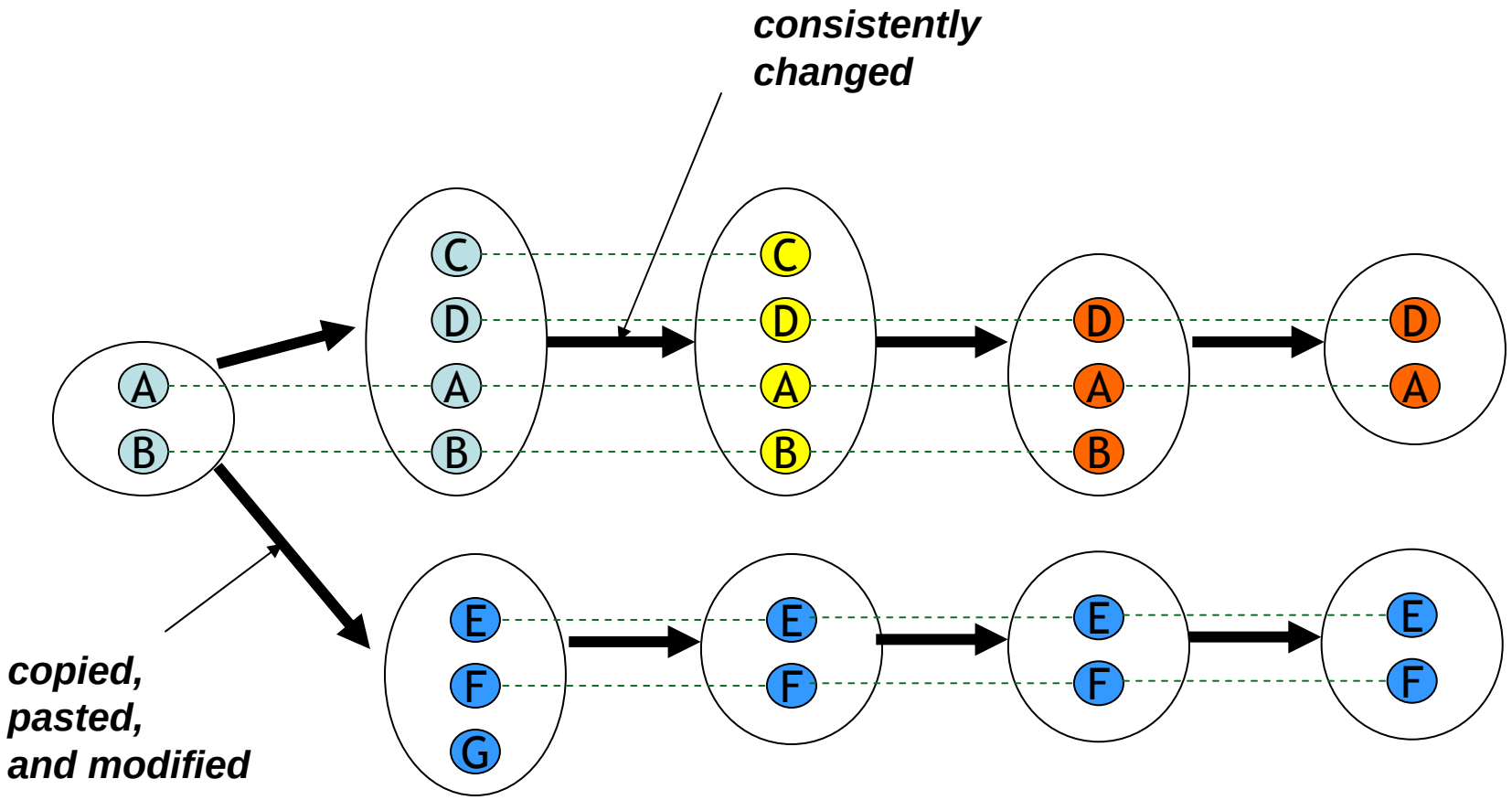
Outline

- ☑ motivation
- ❑ clone genealogy : model and tool
- ❑ potential applications of clone genealogy
- ❑ ongoing work

Model of Clone Evolution

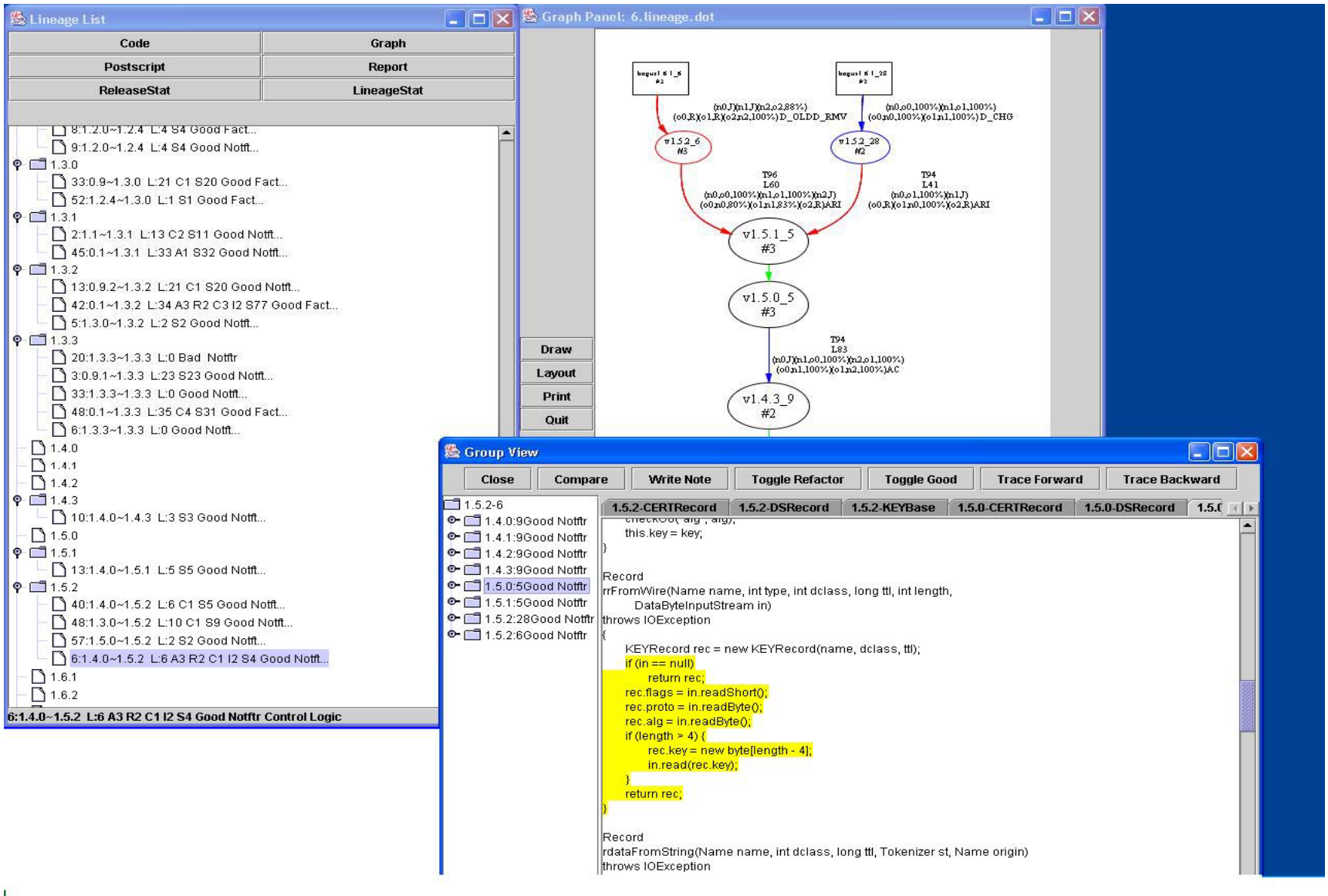


Clone genealogy reflects how programmers evolved code clones by copying, pasting and modifying code.



Clone Genealogy Extractor

- given multiple versions of a program, V_k for $1 \leq k \leq n$.
- find clone groups in each version using a clone detector *(CCFinder)*
- find cloning relationships between clone groups between V_i and V_{i+1}, using the same clone detector
- separate each connected component of cloning relationships, a clone genealogy.
- identify clone evolution patterns in each genealogy.

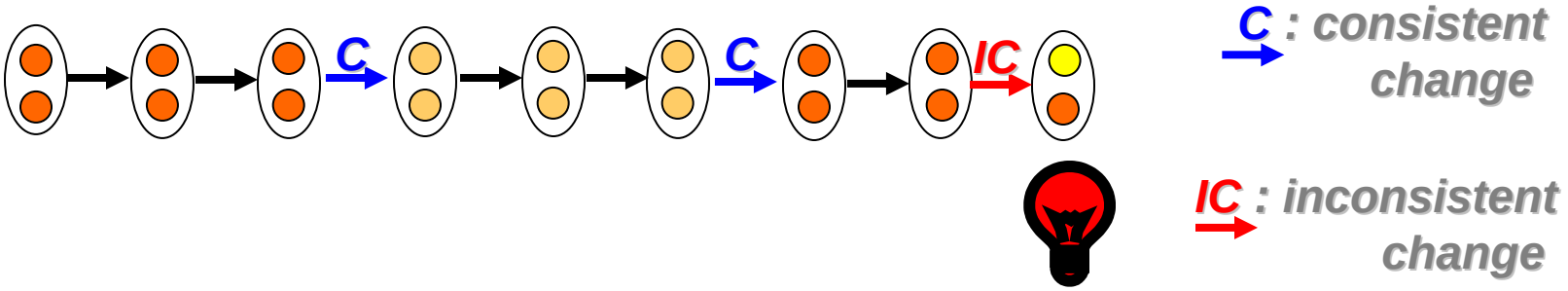

Lineage List
Code
Graph
Postscript
Report
ReleaseStat
LineageStat
8:1.2.0~1.2.4 L:4 S4 Good Fact...
9:1.2.0~1.2.4 L:4 S4 Good Notft...
1.3.0
33:0.9~1.3.0 L:21 C1 S20 Good Fact...
52:1.2.4~1.3.0 L:1 S1 Good Fact...
1.3.1
2:1.1~1.3.1 L:13 C2 S11 Good Notft...
45:0.1~1.3.1 L:33 A1 S32 Good Notft...
1.3.2
13:0.9.2~1.3.2 L:21 C1 S20 Good Notft...
42:0.1~1.3.2 L:34 A3 R2 C3 I2 S77 Good Fact...
5:1.3.0~1.3.2 L:2 S2 Good Notft...
1.3.3
20:1.3.3~1.3.3 L:0 Bad Notftr
3:0.9.1~1.3.3 L:23 S23 Good Notft...
33:1.3.3~1.3.3 L:0 Good Notft...
48:0.1~1.3.3 L:35 C4 S31 Good Fact...
6:1.3.3~1.3.3 L:0 Good Notft...
1.4.0
1.4.1
1.4.2
1.4.3
10:1.4.0~1.4.3 L:3 S3 Good Notft...
1.5.0
1.5.1
13:1.4.0~1.5.1 L:5 S5 Good Notft...
1.5.2
40:1.4.0~1.5.2 L:6 C1 S5 Good Notft...
48:1.3.0~1.5.2 L:10 C1 S9 Good Notft...
57:1.5.0~1.5.2 L:2 S2 Good Notft...
6:1.4.0~1.5.2 L:6 A3 R2 C1 I2 S4 Good Notft...
1.6.1
1.6.2
6:1.4.0~1.5.2 L:6 A3 R2 C1 I2 S4 Good Notftr Control Logic
Graph Panel: 6.lineage.dot
Draw
Layout
Print
Quit
v1.5.2_6
#3
v1.5.2_28
#2
T96
L60
T94
L41
v1.5.1_5
#3
v1.5.0_5
#3
T94
L83
v1.4.3_9
#2
Group View
Close
Compare
Write Note
Toggle Refactor
Toggle Good
Trace Forward
Trace Backward
1.5.2-6
1.4.0:9Good Notftr
1.4.1:9Good Notftr
1.4.2:9Good Notftr
1.4.3:9Good Notftr
1.5.0:5Good Notftr
1.5.1:5Good Notftr
1.5.2:28Good Notftr
1.5.2:6Good Notftr
1.5.2-CERTRecord
1.5.2-DSRecord
1.5.2-KEYBase
1.5.0-CERTRecord
1.5.0-DSRecord
this.key = key;
}
Record
rrFromWire(Name name, int type, int dclass, long ttl, int length,
DataByteInputStream in)
throws IOException
{
KEYRecord rec = new KEYRecord(name, dclass, ttl);
if (in == null)
return rec;
rec.flags = in.readShort();
rec.proto = in.readByte();
rec.alg = in.readByte();
if (length > 4) {
rec.key = new byte[length - 4];
in.read(rec.key);
}
return rec;
}
Record
rdataFromString(Name name, int dclass, long ttl, Tokenizer st, Name origin)
throws IOException

Outline

- ☑motivation
- ☑clone genealogy: model and tool
- ❑potential applications of clone genealogy
- ❑ongoing work

Potential Tool: Warning Cloning Related Bug

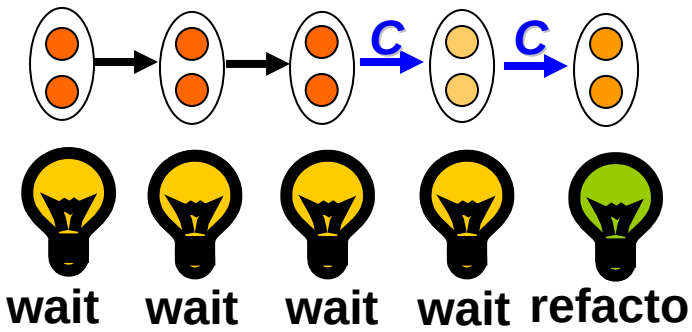
- Failing to update clones consistently often creates a bug in software.
- Goal: warn programmers about inconsistent update of clones.



Potential Tool: Decision Support for Maintaining Clones

- Refactoring too early vs. refactoring too late

- Goal: suggest when to refactor clones.



C : consistent change

Outline

- ☑motivation
- ☑clone genealogy: model and tool
- ☑potential applications of clone genealogy
- ❑ongoing work

Empirical Analysis of Clone Genealogies

- Clone genealogy information *enables* us to study *evolutionary characteristics* of clones.
  - How often do programmers actually make similar changes to clones?
  - What are evolutionary characteristics of clones that are difficult to refactor?
  - How long do clones survive in the system before they disappear?

Summary

- We have built a tool that extracts history of code clones from a set of program versions.
- We envision that clone genealogy information can be used for potential clone maintenance tools.
- Our previous & ongoing work checks the validity of conventional wisdom using a clone genealogy extractor.

Acknowledgement:
Software engineering laboratory at Osaka University provided CCFinder.
GRASE research group at UCSC provided Kenyon.
Collaborators: Vibha Sazawal and Gail Murphy

University of Washington
Computer Science & Engineering