Sub-clones: Considering the Part Rather than the Whole

Robert Tairas
Department of Computer and Information Sciences
University of Alabama at Birmingham

Jeff Gray
Department of Computer Science
University of Alabama

University of Alabama at Birmingham

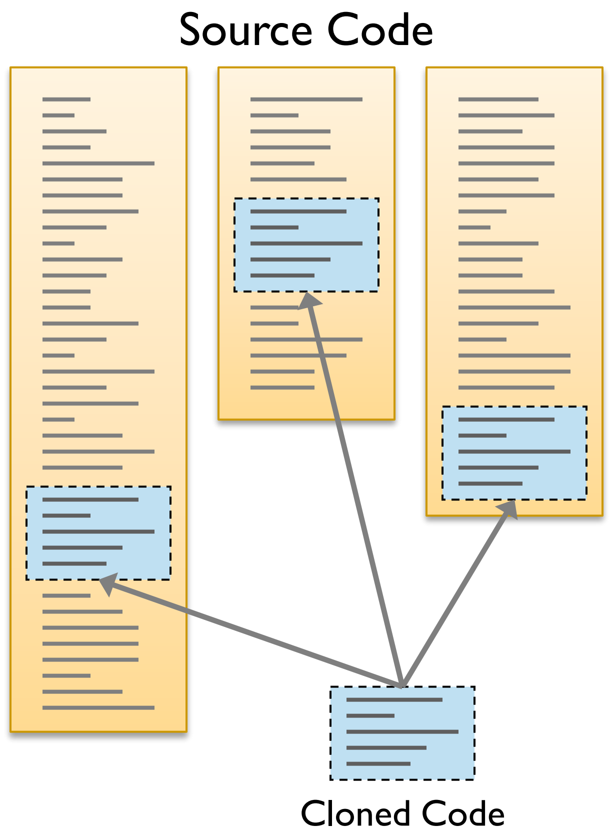
Software Composition and Modeling Lab

University of Alabama

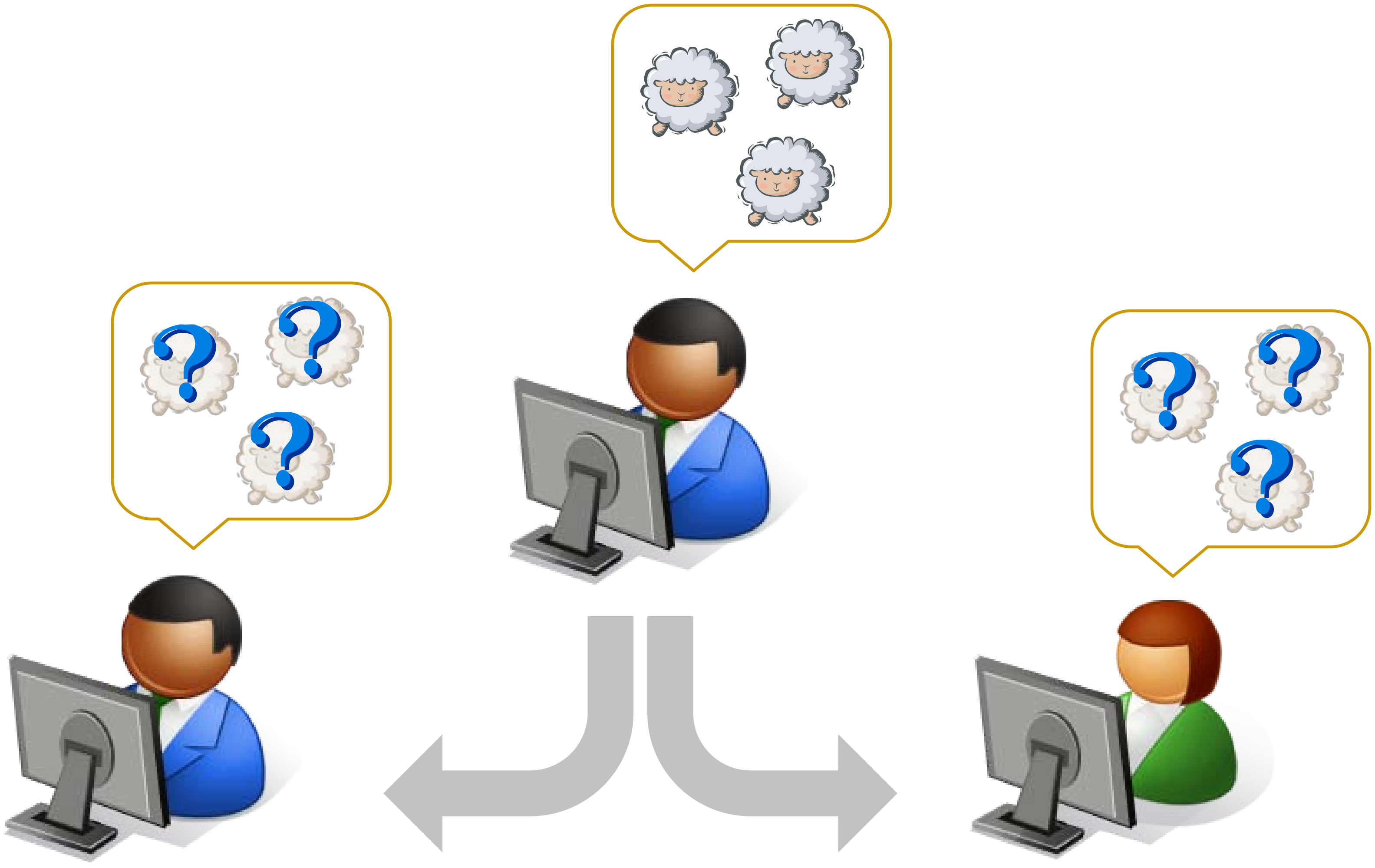
This research is supported by
NSF grant CPA-0702764

Cloning in Software

- Code Clones:
  - A section of code that is duplicated in multiple locations in a program
- Different granularity levels:
  - Statements, Block, Method, Class, Program
- Clone Group:
  - Clones of the same duplication



Maintaining Clones

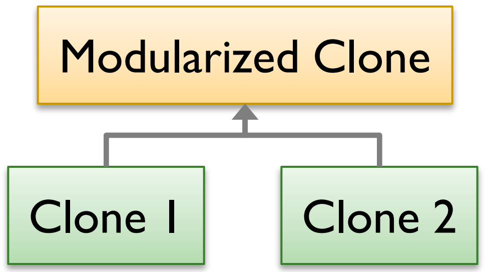


After a period of time

A new programmer

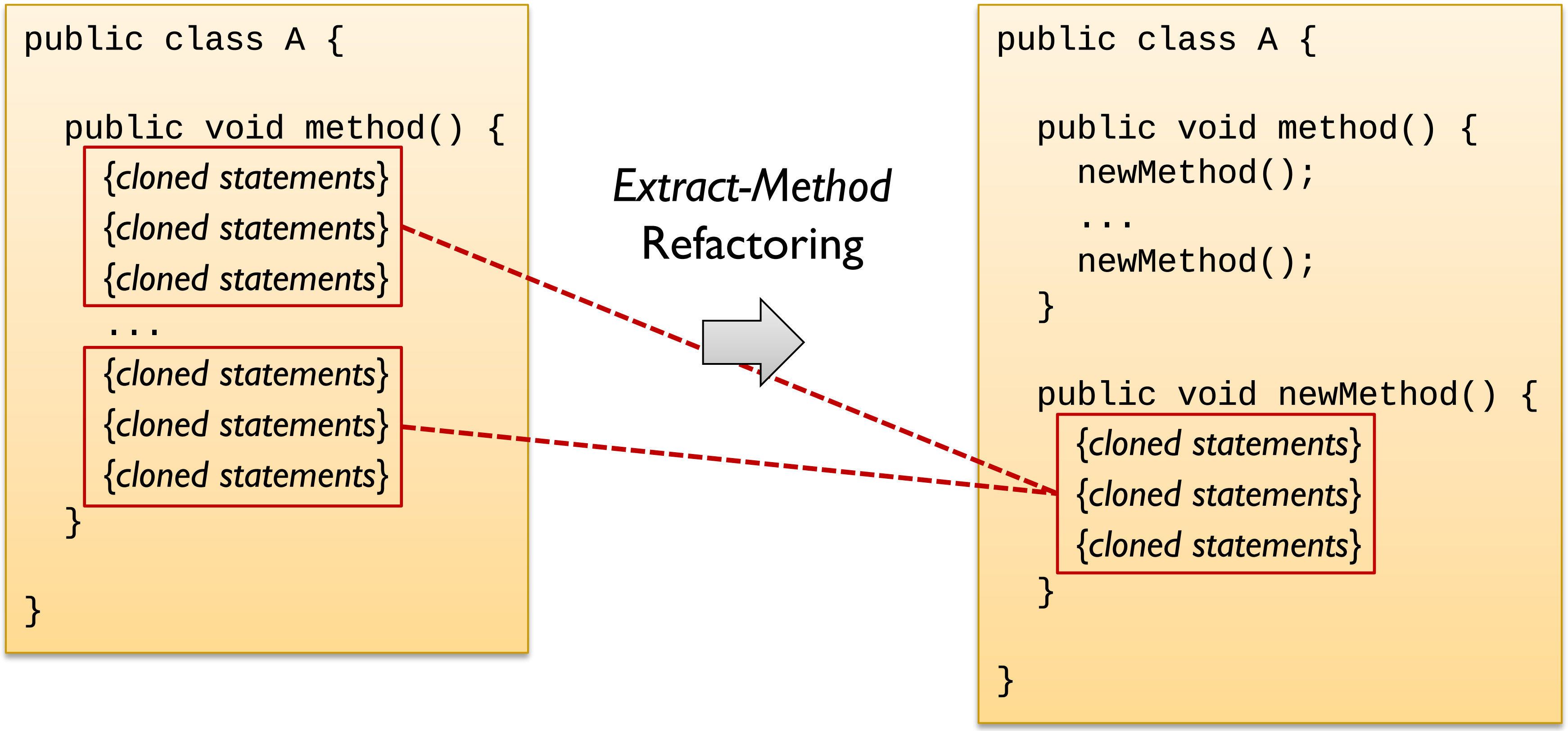
Removing Clones through Refactoring

- Modularizing the code represented by clones through appropriate abstractions may improve code quality
  - Less duplicated code to maintain
  - Ease of future maintenance efforts



- *Refactoring* is one means of improving the quality of code
  - The goal of refactoring is to preserve the external behavior of code while improving its internal structure

Refactoring Clones

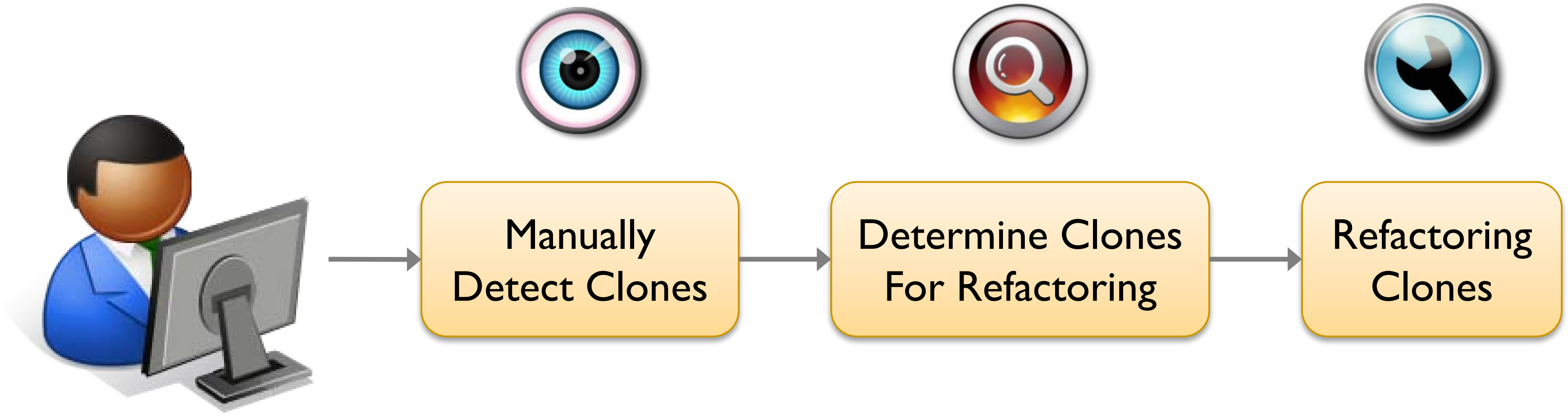


```
public class A {

  public void method() {
    {cloned statements}
    {cloned statements}
    {cloned statements}
    ...
    {cloned statements}
    {cloned statements}
    {cloned statements}
  }

}
```

Extract-Method Refactoring

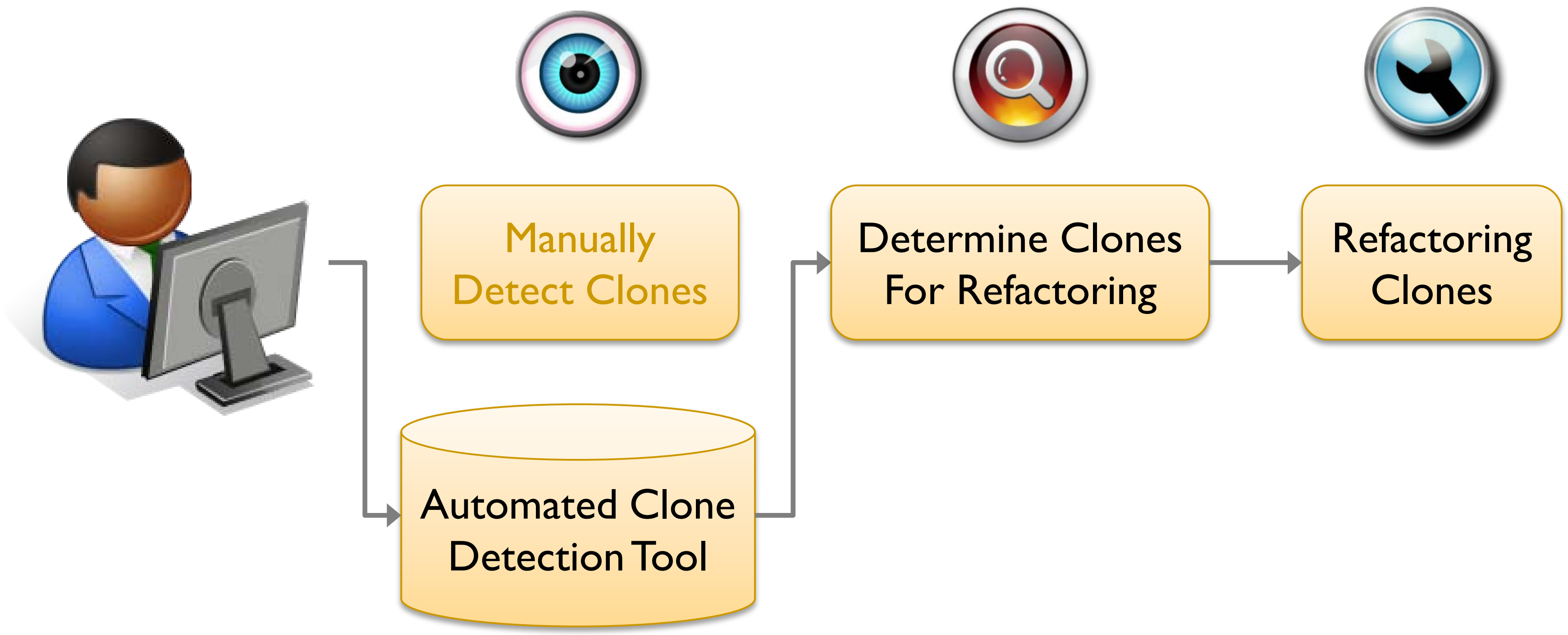
```
public class A {

  public void method() {
    newMethod();
    ...
    newMethod();
  }

  public void newMethod() {
    {cloned statements}
    {cloned statements}
    {cloned statements}
  }

}
```

Clone Refactoring Process



- Changes between two versions
  - First version contains original code
  - Second version contains refactored code

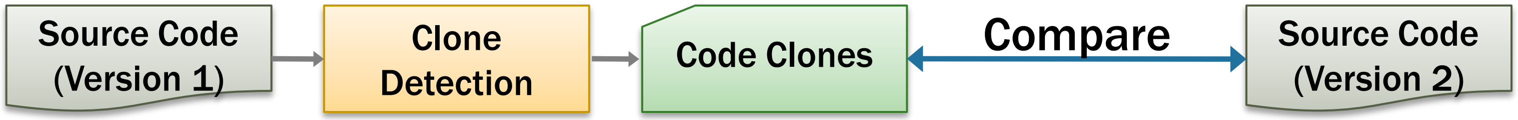
Clone Refactoring Process


Manually Detect Clones
Automated Clone Detection Tool
Determine Clones For Refactoring
Refactoring Clones

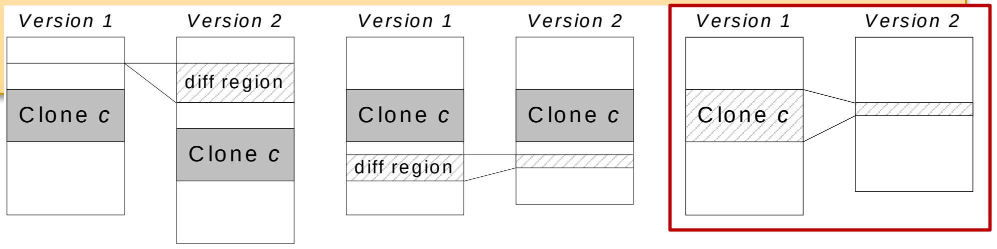
- What are the refactoring characteristics of clones detected by a clone detection tool, if such a tool was used in the clone maintenance process?

Approach: Observing Refactorings

- Observing actual clone-related refactorings in multiple release versions of JBoss (v2.2.0–4.2.3)
  - Used Simian clone detection tool



```
 @@ -2471,13 +2469,7 @@
                 scan_position.current_slot = Page.INVALID_SLOT_NUMBER;

                 // release the scan lock now that we have saved away the row.
-
-                if (scan_position.current_scan_pageno != 0)
-                {
-                   this.getLockingPolicy().unlockScan(
-                      scan_position.current_scan_pageno);
-                   scan_position.current_scan_pageno = 0;
-                }
+                unlockCurrentScan(scan_position);
             }
         }
     }
```



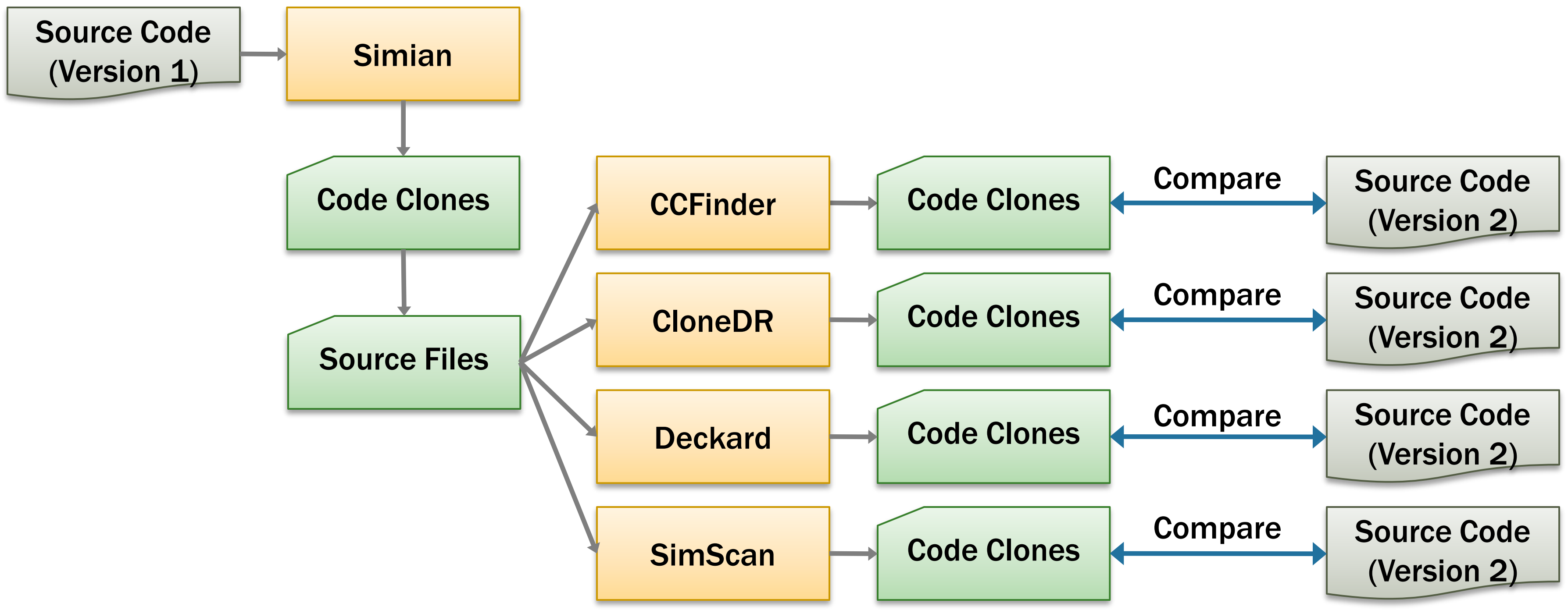
Refactoring of Simian Clones

- Observations
  - 21 *Extract Method*-type Refactorings
  - Range of refactored code not equal to the range reported as a clone

Type	Total
Extract Method	14
Extract Method with *Pull-up Method*	1
Extract Method to utility class	6
Total	21

Observing with Other Tools

- Consider clones reported by other tools
  - CCFinder, CloneDR, Deckard, and SimScan
- Run these tools on source files associated with the 21 *Extract Method*-type refactorings from Simian clones



Evaluation: Tool Coverage

- Coverage of 21 *Extract Method*-type refactorings in JBoss
  - Initially detected by using Simian clones
- Reported clones that exactly covered the refactored code were less than half for all the tools

Tool	Exact Coverage	Larger Coverage
1. CCFinder	4 (19%)	8 (38%)
2. CloneDR	6 (28%)	9 (42%)
3. Deckard	8 (38%)	3 (14%)
4. Simian	2 (9%)	0 (0%)
5. Simscan	6 (28%)	12 (57%)

Refactoring in Clone Ranges

```
1 2   4 5   protected String getValue(String name, String value) {
1 2   4 5     if (value.startsWith("${") && value.endsWith("}")) {
1 2 3 4 5 -     try {
1 2 3 4 5 -       String propertyName = value.substring(2, value.length()-1);
1 2 3 4 5 -       ObjectName propertyServiceON = new ObjectName(“...");
1 2 3 4 5 -       KernelAbstraction kernelAbstraction = KernelAbstractionFactory.getInstance();
1 2 3 4 5 -       String propertyValue = (String)kernelAbstraction.invoke(...);
1 2 3   5 -       log.debug("Replaced ejb-jar.xml element " + name + " with value " + propertyValue);
1 2 3   5 -       return propertyValue;
1 2 3   5 -     } catch (Exception e) {
1 2 3   5 -       log.warn("Unable to look up property service for ejb-jar.xml element " + ...);
1 2 3   5 -     }
          +     String replacement = StringPropertyReplacer.replaceProperties(value);
          +     if (replacement != null)
          +       value = replacement;
1 2     5     }
1 2     5     return value;
1 2     5   }
```

```
   if (edge instanceof MTransition) {
      MTransition tr = (MTransition) edge;
-     FigTrans trFig = new FigTrans(tr);
-     // set source and dest
-     // set any arrowheads, labels, or colors
-     MStateVertex sourceSV = tr.getSource();
-     MStateVertex destSV = tr.getTarget();
-     FigNode sourceFN = (FigNode) lay...
-     FigNode destFN = (FigNode) lay...
-     trFig.setSourcePortFig(sourceFN);
-     trFig.setSourceFigNode(sourceFN);
-     trFig.setDestPortFig(destFN);
-     trFig.setDestFigNode(destFN);
+     FigTrans trFig = new FigTrans(tr, lay);
      return trFig;
   }
```

- Refactoring performed on only part of the reported clone range
  - Sub-clone refactoring

Evaluation: Focus on Deckard

- Deckard selected due to tree-based tool performance
  - JBoss re-evaluated
  - Additional artifacts: ArgoUML (v0.10.1–0.26) and Apache Derby (v10.1.1.0–10.5.3.0)

Property		JBoss	ArgoUML	Derby
Refactoring Coverage	Exact coverage	19	17	12
	Sub-clone coverage	14	9	15
Coverage Levels	Same level	4	4	6
	1 level above	9	2	8
	> 1 level above	1	3	1
Clone Differences	Refactorable	7	4	8
	Not Refactorable	7	5	7

Evaluation: Focus on Deckard

- Reported clone range mainly the same level or one syntactic level above the actual refactored code
  - Possibly to keep some logic in the original location

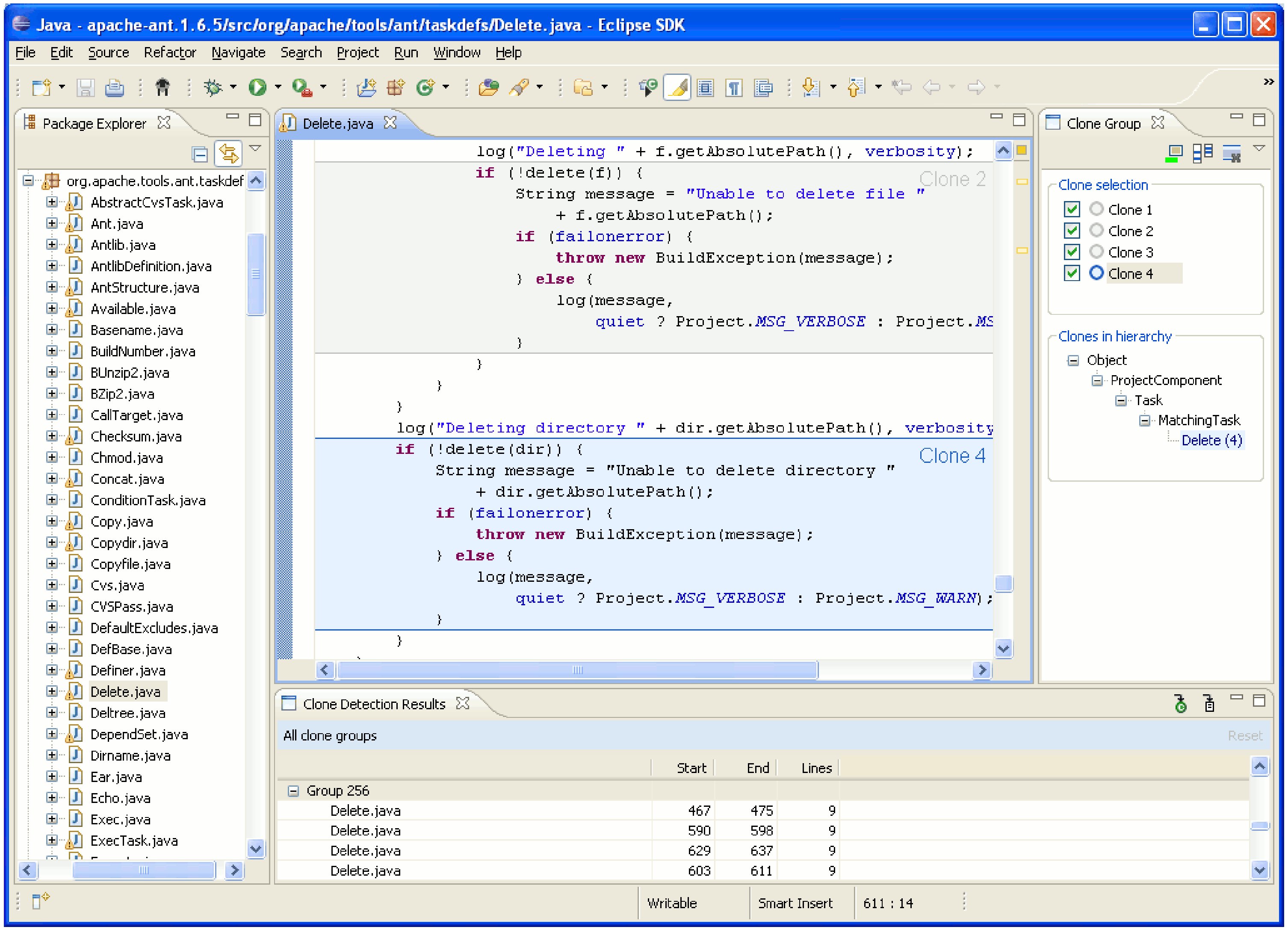
Property		JBoss	ArgoUML	Derby
Refactoring Coverage	Exact coverage	19	17	12
	Sub-clone coverage	14	9	15
Coverage Levels	Same level	4	4	6
	1 level above	9	2	8
	> 1 level above	1	3	1
Clone Differences	Refactorable	7	4	8
	Not Refactorable	7	5	7

- Programmers only refactored a sub-clone even when the entire clone was refactorable

Conclusion

- We observed the actual refactoring of clones by evaluating source code changes between multiple versions
  - In various instances only part of the reported clone (i.e., sub-clone) was refactored
- We conclude that sub-clone refactoring should be included in the clone maintenance process
- Future Work
  - Individual evaluation of other clone detection tools
  - Provide support for sub-clone refactoring in an IDE

CeDAR plug-in


Java - apache-ant.1.6.5/src/org/apache/tools/ant/taskdefs/Delete.java - Eclipse SDK
File Edit Source Refactor Navigate Search Project Run Window Help
Package Explorer
Delete.java
Clone Group
Clone Detection Results
Reset

org.apache.tools.ant.taskdef
- AbstractCvsTask.java
- Ant.java
- Antlib.java
- AntlibDefinition.java
- AntStructure.java
- Available.java
- Basename.java
- BuildNumber.java
- BUnzip2.java
- BZip2.java
- CallTarget.java
- Checksum.java
- Chmod.java
- Concat.java
- ConditionTask.java
- Copy.java
- Copydir.java
- Copyfile.java
- Cvs.java
- CVSPass.java
- DefaultExcludes.java
- DefBase.java
- Definer.java
- Delete.java
- Deltree.java
- DependSet.java
- Dirname.java
- Ear.java
- Echo.java
- Exec.java
- ExecTask.java

```
                log("Deleting " + f.getAbsolutePath(), verbosity);
                if (!delete(f)) {                                        Clone 2
                    String message = "Unable to delete file "
                        + f.getAbsolutePath();
                    if (failonerror) {
                        throw new BuildException(message);
                    } else {
                        log(message,
                            quiet ? Project.MSG_VERBOSE : Project.MS
                    }
                }
            }
        }
        log("Deleting directory " + dir.getAbsolutePath(), verbosity
        if (!delete(dir)) {                                              Clone 4
            String message = "Unable to delete directory "
                + dir.getAbsolutePath();
            if (failonerror) {
                throw new BuildException(message);
            } else {
                log(message,
                    quiet ? Project.MSG_VERBOSE : Project.MSG_WARN);
            }
        }
```

Clone selection
- ✓ ○ Clone 1
- ✓ ○ Clone 2
- ✓ ○ Clone 3
- ✓ ● Clone 4

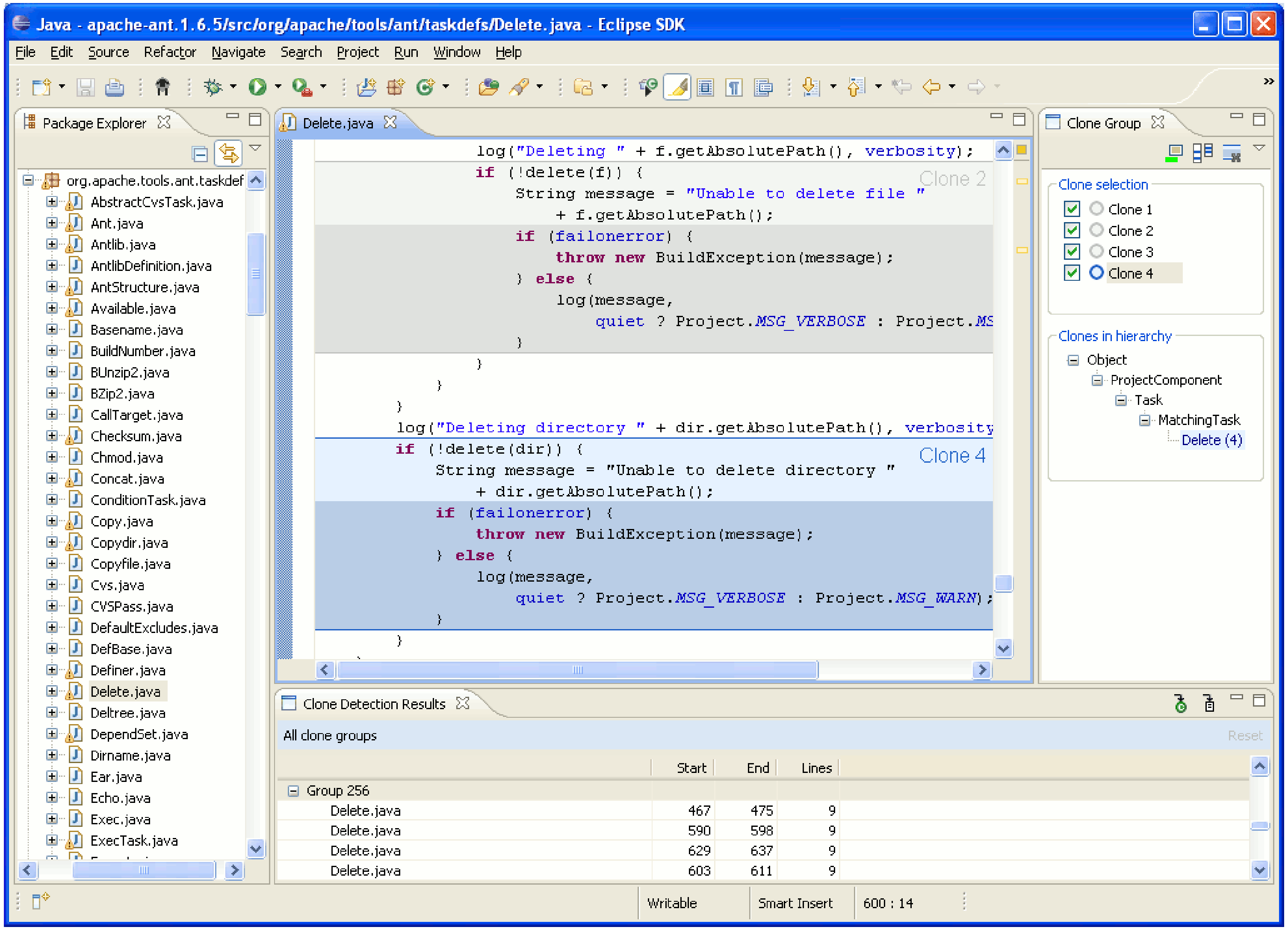
Clones in hierarchy
- Object
  - ProjectComponent
    - Task
      - MatchingTask
        - Delete (4)

All clone groups

	Start	End	Lines
Group 256			
Delete.java	467	475	9
Delete.java	590	598	9
Delete.java	629	637	9
Delete.java	603	611	9

Writable   Smart Insert   611 : 14

Sub-clones in CeDAR


Java - apache-ant.1.6.5/src/org/apache/tools/ant/taskdefs/Delete.java - Eclipse SDK
File Edit Source Refactor Navigate Search Project Run Window Help
Package Explorer
Delete.java
Clone Group
Clone Detection Results
Writable
Smart Insert
600 : 14

Package Explorer:

- org.apache.tools.ant.taskdef
  - AbstractCvsTask.java
  - Ant.java
  - Antlib.java
  - AntlibDefinition.java
  - AntStructure.java
  - Available.java
  - Basename.java
  - BuildNumber.java
  - BUnzip2.java
  - BZip2.java
  - CallTarget.java
  - Checksum.java
  - Chmod.java
  - Concat.java
  - ConditionTask.java
  - Copy.java
  - Copydir.java
  - Copyfile.java
  - Cvs.java
  - CVSPass.java
  - DefaultExcludes.java
  - DefBase.java
  - Definer.java
  - Delete.java
  - Deltree.java
  - DependSet.java
  - Dirname.java
  - Ear.java
  - Echo.java
  - Exec.java
  - ExecTask.java

Delete.java:

```
                log("Deleting " + f.getAbsolutePath(), verbosity);
                if (!delete(f)) {                                    Clone 2
                    String message = "Unable to delete file "
                        + f.getAbsolutePath();
                    if (failonerror) {
                        throw new BuildException(message);
                    } else {
                        log(message,
                            quiet ? Project.MSG_VERBOSE : Project.MS
                    }
                }
            }
        }
        log("Deleting directory " + dir.getAbsolutePath(), verbosity
        if (!delete(dir)) {                                          Clone 4
            String message = "Unable to delete directory "
                + dir.getAbsolutePath();
            if (failonerror) {
                throw new BuildException(message);
            } else {
                log(message,
                    quiet ? Project.MSG_VERBOSE : Project.MSG_WARN);
            }
        }
```

Clone Group:

Clone selection
- ✓ ○ Clone 1
- ✓ ○ Clone 2
- ✓ ○ Clone 3
- ✓ ● Clone 4

Clones in hierarchy
- Object
  - ProjectComponent
    - Task
      - MatchingTask
        - Delete (4)

Clone Detection Results

All clone groups — Reset

	Start	End	Lines
Group 256			
Delete.java	467	475	9
Delete.java	590	598	9
Delete.java	629	637	9
Delete.java	603	611	9

Thank you

- Personal:
  - http://www.cis.uab.edu/tairasr
- Code Clones Literature:
  - http://www.cis.uab.edu/tairasr/clones/literature
- SoftCom Laboratory:
  - http://www.cis.uab.edu/softcom