Automated Transplantation and Differential Testing for Clones

Tianyi Zhang, Miryung Kim
University of California, Los Angeles

Problem Statement

- Code clones are common in modern software systems.
- Developers often find it difficult to examine the runtime behavior of clones.
- This problem is exacerbated by a lack of tests. 46% of clone pairs are only *partially* covered by existing test suites.
- We present Grafter to reuse tests between clones and examine behavior differences.

A pair of similar but not identical clones that are detected by an existing clone detection tool, Deckard [ICSE 2007].

```
public class Copy extends Task{
  private IncludePatternSet  includes;
  public void setIncludes(String patterns){
    …
    StringTokenizer tok = new StringTokenizer(patterns, “,”);
    while(tok.hasMoreTokens()){
       includes.addPattern(tok.next);
    }
    …
  }
}
class IncludePatternSet {
    public Set<String> set;
    public void addPattern(String s) { set.add(s); };
}
```

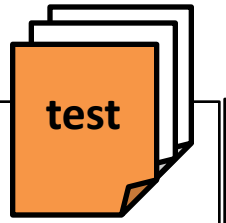
Copy.java

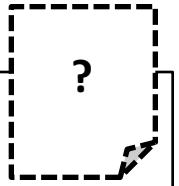
```
public class Delete extends Task{
  private ExcludePatternSet  excludes;
  public void setExcludes(String patterns){
    …
    StringTokenizer tok = new StringTokenizer(patterns, “,”);
     while(tok.hasMoreTokens()){
       excludes.addPattern(tok.next);
     }
     …
  }
}
class ExcludePatternSet {
    public Set<String> set;
    public void addPattern(String s) { set.add(s); };
}
```

Delete.java

* The example is adapted from Apache Ant 1.9.6 for presentation purposes .

A programmer updates the use of StringTokenizer to StringUtils.split in the Copy and Delete classes.






```
public class Copy extends Task{
  private IncludePatternSet includes;
  public void setIncludes(String patterns){
  …
+  String[] tokens = StringUtils.split(patterns, “,”);
+  for(String tok : tokens){
+      includes.addPattern(tok);
+  }
  …
  }
}
```

Copy.java

```
public class Delete extends Task{
  private ExcludePatternSet excludes;
  public void setExcludes(String patterns){
  …
+  String[] tokens = StringUtils.split(patterns, “.”);
+  for(String tok : tokens){
+      excludes.addPattern(tok);
+  }
  …
  }
}
```

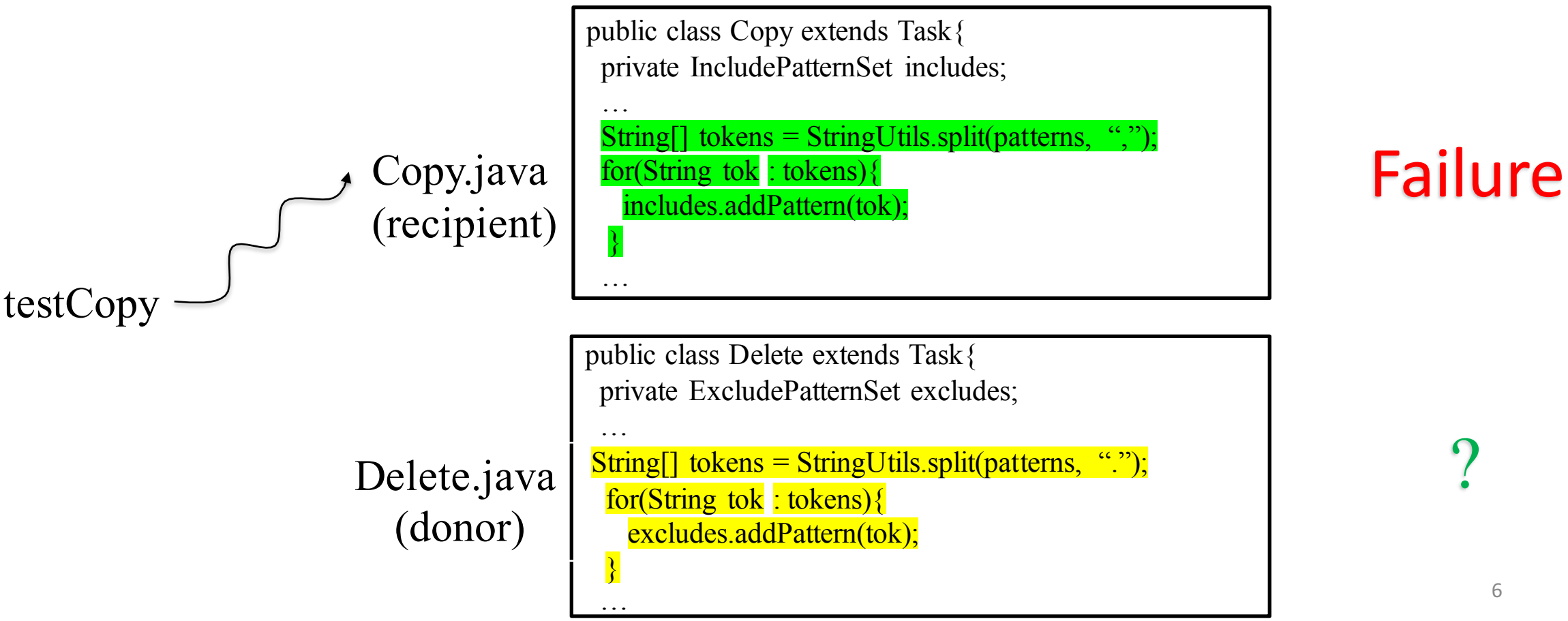
Delete.java

* The example is adapted from Apache Ant 1.9.6 for presentation purposes .

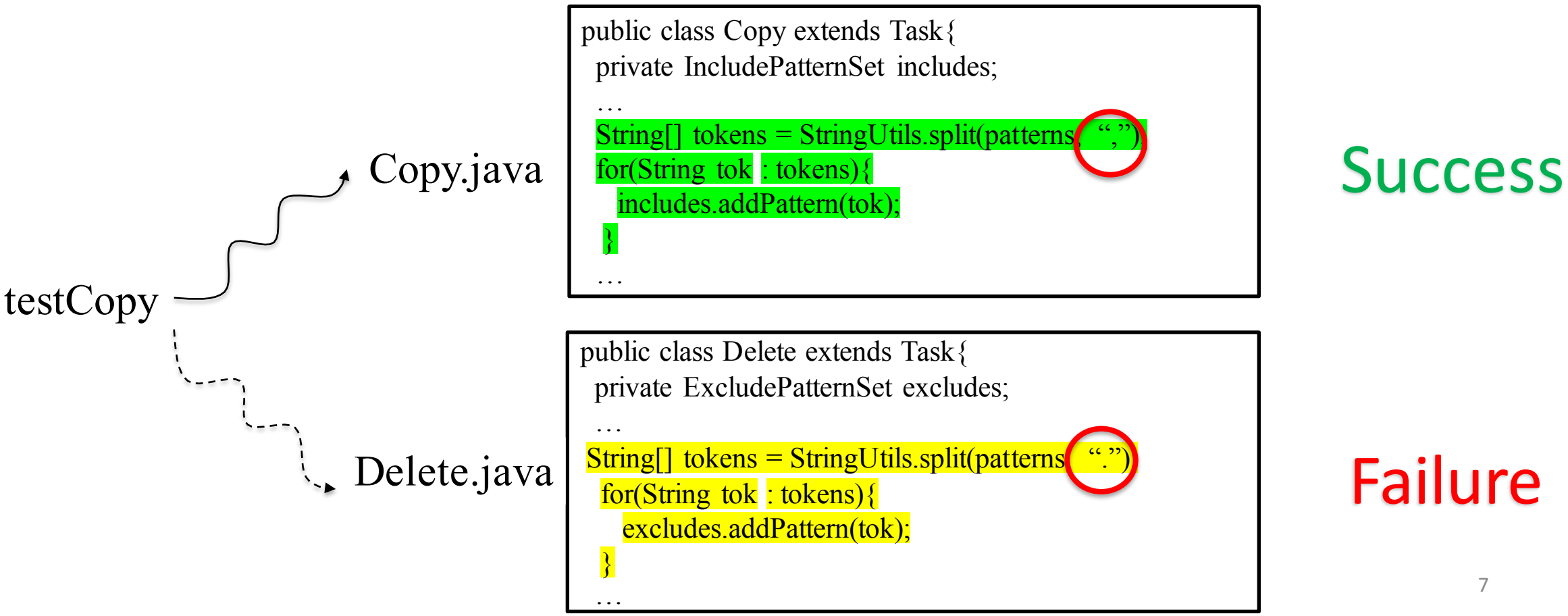
Limitation of Existing Techniques

- Existing test reuse technique for clones works only at a method or class level and requires a reuse plan. [Makady & Walker]
- Existing differential testing or random testing techniques are not geared towards **intra method clones** [Geno, Diffut, Randoop]
- Existing clone inconsistency detection techniques do not detect behavioral differences between clones [Jiang et al., CBCD, SPA]

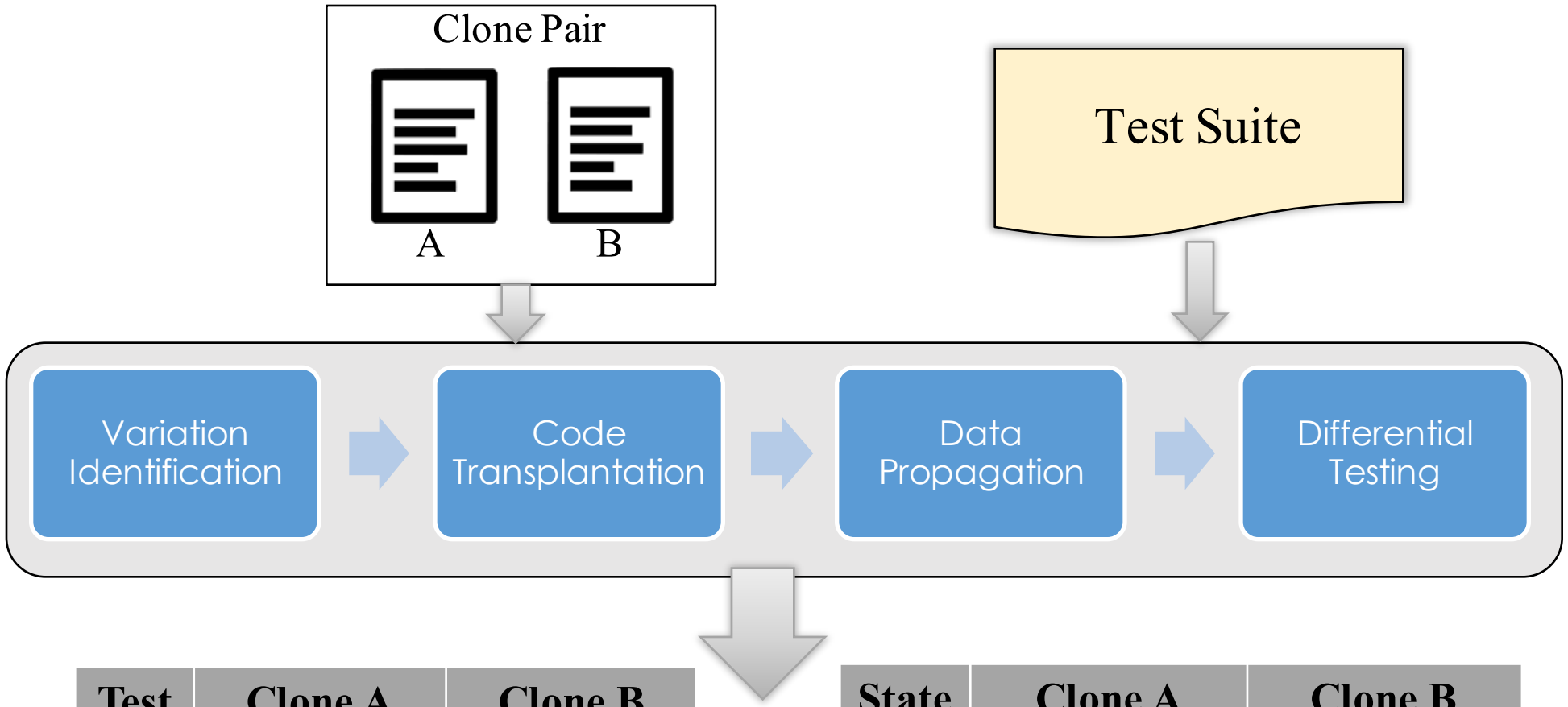
Grafter: Automated Test Reuse and Differential Testing



Grafter: Automated Test Reuse and Differential Testing



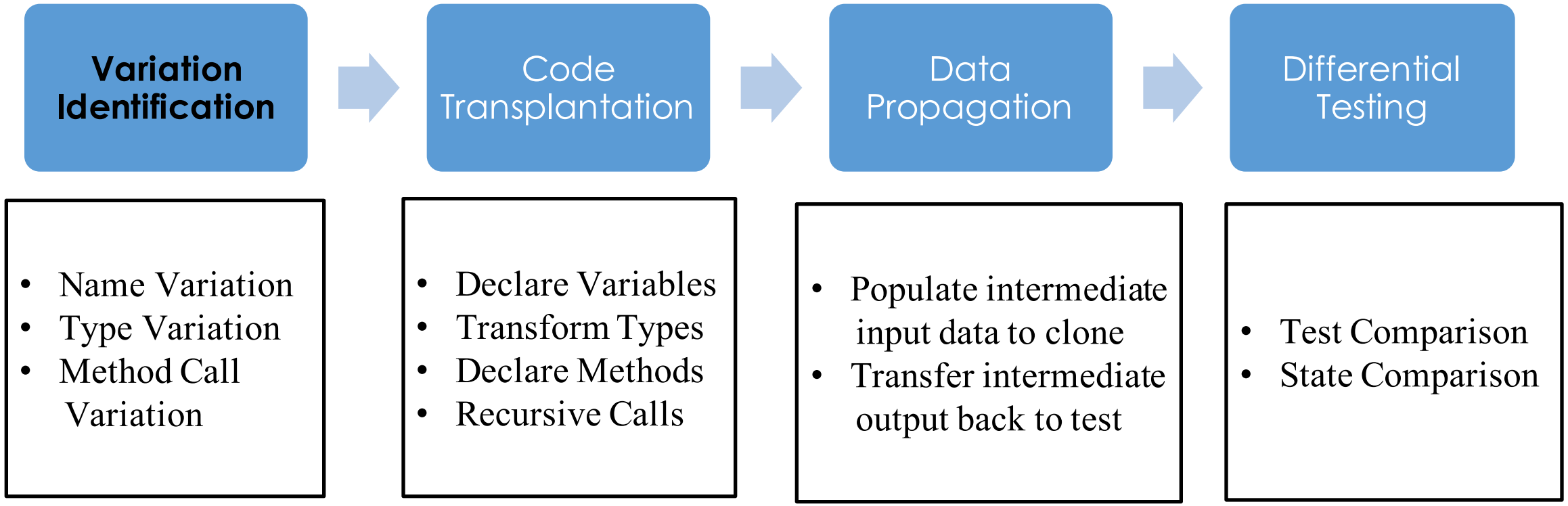
Grafter Approach Overview



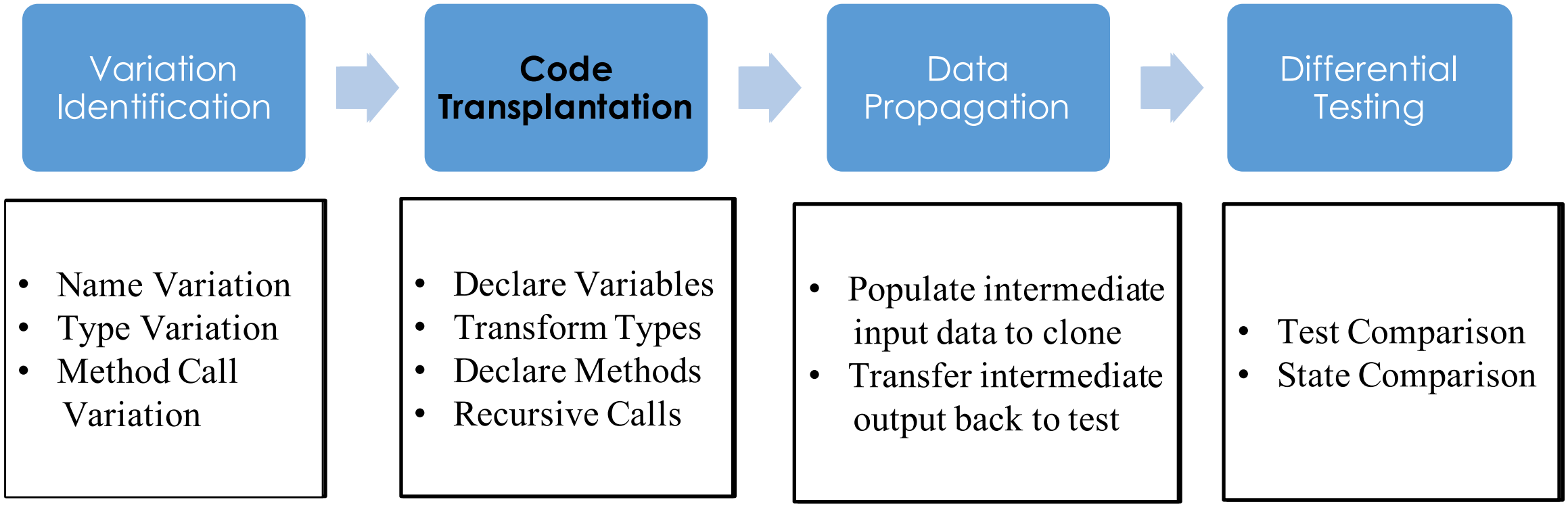
Test	Clone A	Clone B
T1	pass	fail
T2	pass	pass

State	Clone A	Clone B
S1	“string”	“string”
S2	true	false

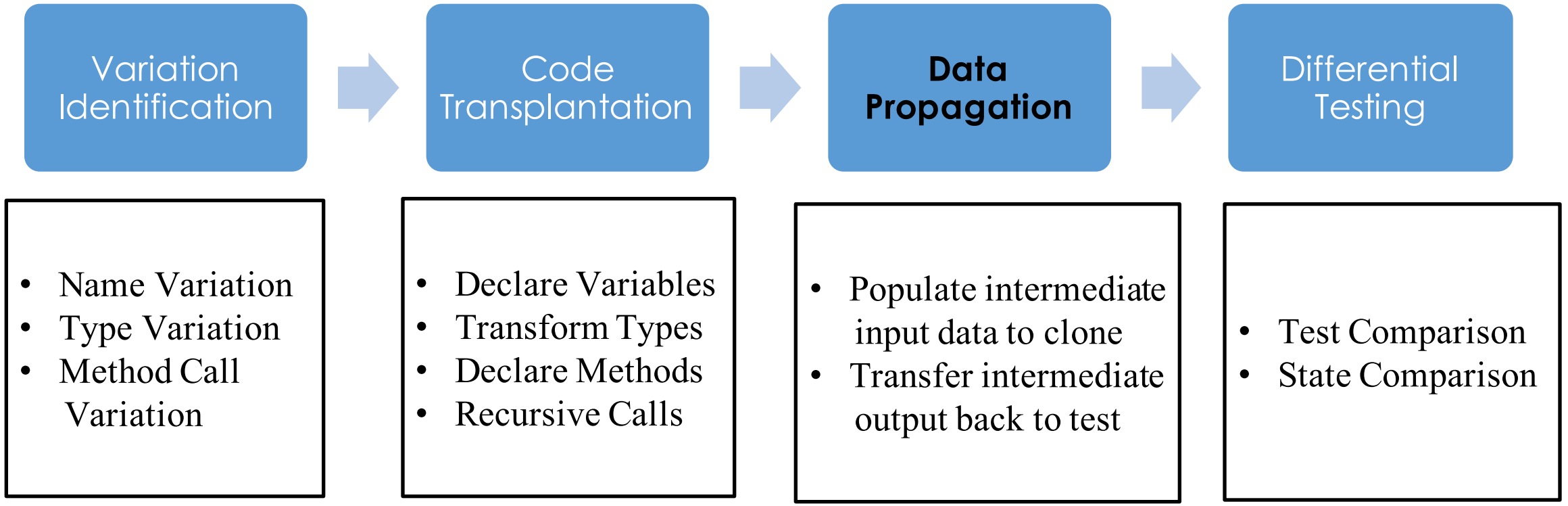
Grafter Approach Overview



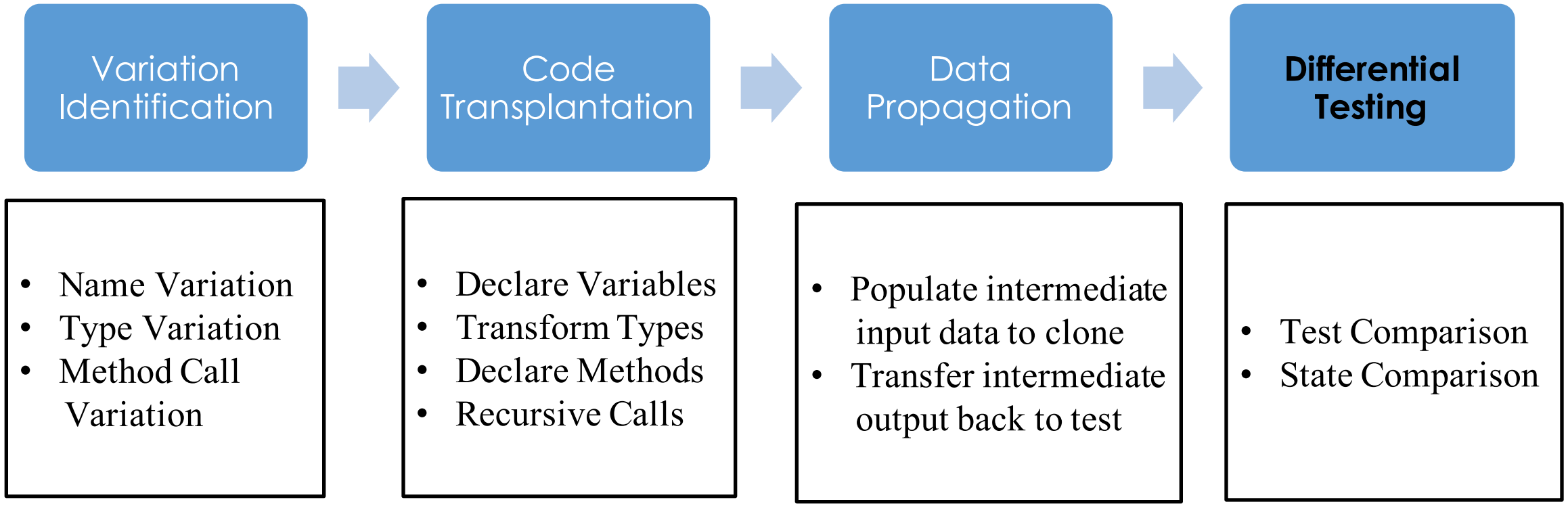
Grafter Approach Overview



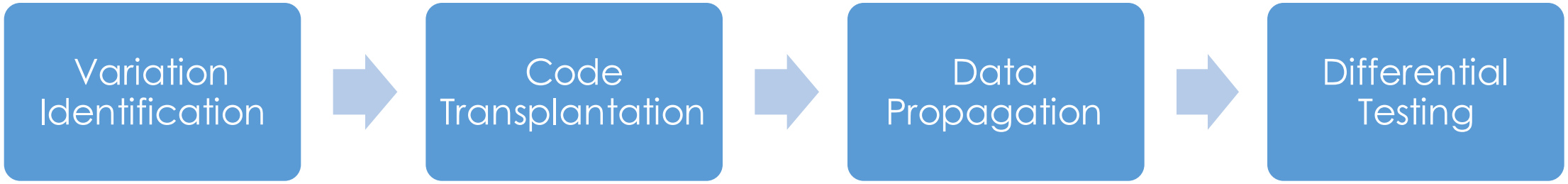
Grafter Approach Overview



Grafter Approach Overview



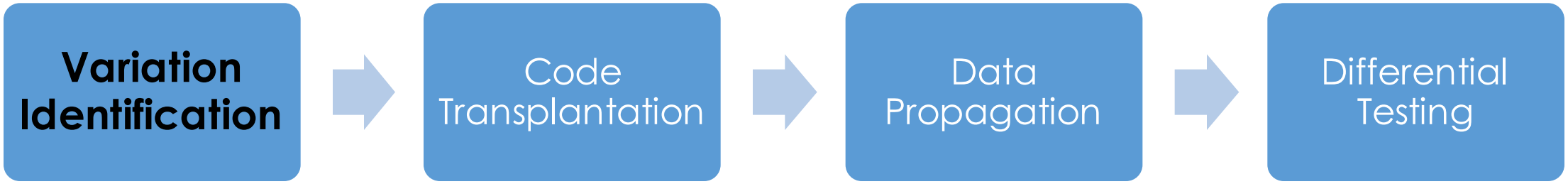
Step 1: Variation Identification




```
public class Copy extends Task{
   private IncludePatternSet includes;
   public void setIncludes(String patterns){
      …
     String[] tokens = StringUtils.split(patterns, “,”);
     for(String tok : tokens){
        includes.addPattern(tok);
     }
   }
   …
```

```
public class Delete extends Task{
   private ExcludePatternSet excludes;
   public void setExcludes(String patterns){
      …
     String[] tokens = StringUtils.split(patterns, “.”);
     for(String tok : tokens){
        excludes.addPattern(tok);
     }
   }
   …
```

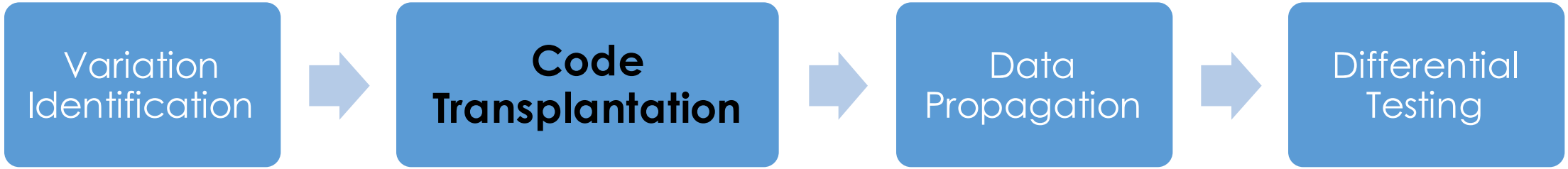
Step 1: Variation Identification




```
public class Copy extends Task{
   private IncludePatternSet  includes;
   public void setIncludes(String patterns){
      …
     String[] tokens = StringUtils.split(patterns, “,”);
     for(String tok : tokens){
        includes.addPattern(tok);
     }
   }
   …
```

```
public class Delete extends Task{
   private ExcludePatternSet excludes;
   public void setExcludes(String patterns){
      …
     String[] tokens = StringUtils.split(patterns, “.”);
     for(String tok : tokens){
        excludes.addPattern(tok);
     }
   }
   …
```

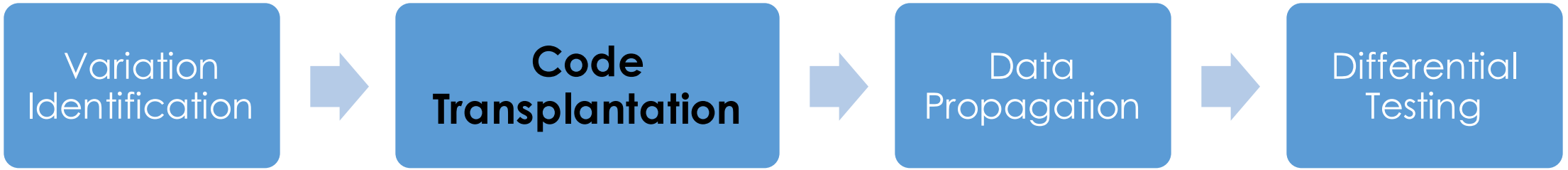
Step 2: Code Transplantation




```
public class Copy extends Task{
   private IncludePatternSet includes;
   public void setIncludes(String patterns){
      …
     String[] tokens = StringUtils.split(patterns, “,”);
     for(String tok : tokens){
        includes.addPattern(tok);
     }
   }
   …
```

```
public class Delete extends Task{
   private ExcludePatternSet excludes;
   public void setExcludes(String patterns){
      …
 String[] tokens = StringUtils.split(patterns, “.”);
      for(String tok : tokens){
         excludes.addPattern(tok);
      }
   }
   …
```

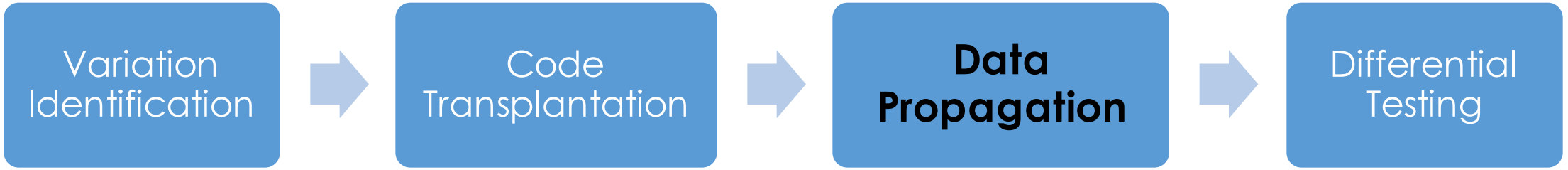
Step 2: Code Transplantation




```
public class Copy extends Task{
   private IncludePatternSet includes;
+  private ExcludePatternSet excludes;
   public void setIncludes(String patterns){
      …
+     excludes = new ExcludePatternSet();
      String[] tokens = StringUtils.split(patterns, ".");
      for(String tok : tokens){
         excludes.addPattern(tok);
      }
   }
   …
```

```
public class Delete extends Task{
   private ExcludePatternSet excludes;
   public void setExcludes(String patterns){
      …
      String[] tokens = StringUtils.split(patterns, ".");
      for(String tok : tokens){
         excludes.addPattern(tok);
      }
   }
   …
```

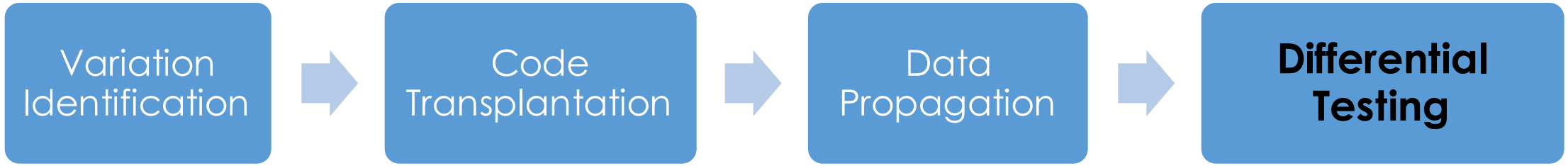
Step 3: Data Propagation




```
public class Copy extends Task{
   private IncludePatternSet includes;
+  private ExcludePatternSet excludes;
   public void setIncludes(String patterns){
      …
+     excludes = new ExcludePatternSet();
+     excludes.set = includes.set;
      String[] tokens = StringUtils.split(patterns, ".");
      for(String tok : tokens){
         excludes.addPattern(tok);
      }
+     includes.set = excludes.set;
   }
   …
```

```
public class Delete extends Task{
   private ExcludePatternSet excludes;
   public void setExcludes(String patterns){
      …
      String[] tokens = StringUtils.split(patterns, ".");
      for(String tok : tokens){
         excludes.addPattern(tok);
      }
   }
   …
```

Step 4: Differential Testing



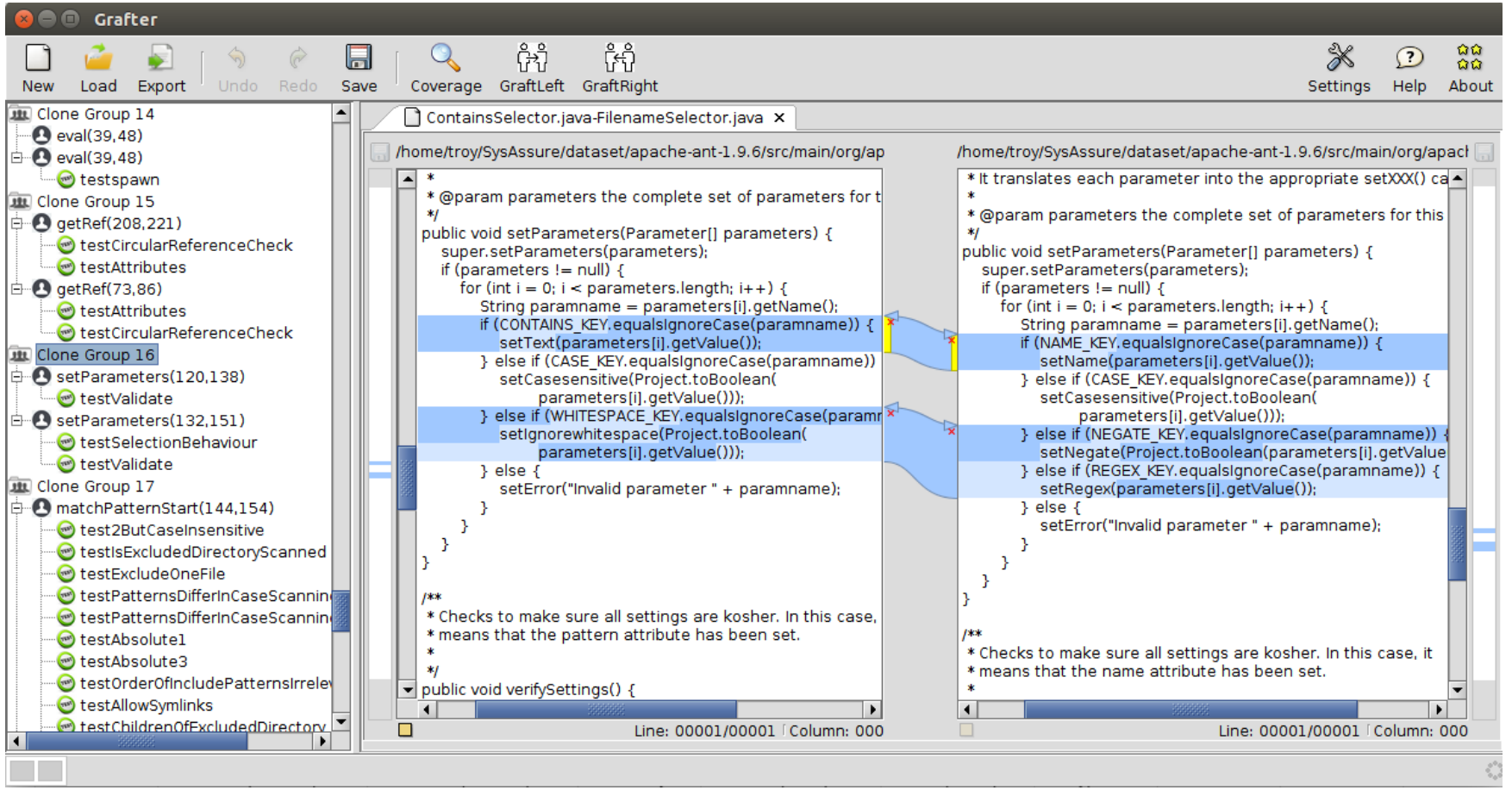
Test	Copy.java	Delete.java
testCopy	pass	fail

Test-level Comparison

State	Copy.java	Delete.java
patterns	"src/*.java, test/*.java"	"src/*.java, test/*.java"
tokens	["src/*.java", "test/*.java"]	["src/*", "java, test/*", "java"]
in(ex)cludes	<IncludePatternSet> <set> ["src/*.java", "test/*.java"] </set> </IncludePatternSet>	<ExcludePatternSet> <set> ["src/*", "java, test/*", "java"] </set> </ExcludePatternSet>

State-level Comparison

- Our tool & dataset are now publicly available.



Tool & Dataset: http://web.cs.ucla.edu/~tianyi.zhang/grafter.html

- Behavioral differences are represented in tables and highlighted for ease of investigation.

Test-level Comparison

Test	ContainsSelector(120 , 138)	FilenameSelector(13...	Comparison
ContainsSelectorTest.testValidate	true	true	
FilenameSelectorTest.testSelectionBehav...	false	true	
FilenameSelectorTest.testValidate	true	true	

State-level Comparison

State - ContainsSelector(...	Value	State - FilenameSelector...	Value	Comparison
this.errmsg	<string>Invalid parameter ...	this.errmsg	<null/>	
i	no value	i	no value	
this.contains	<null/>	this.negated	<boolean>false</boolean>	
this.ignorewhitespace	no value	this.negated	<boolean>false</boolean>	
this.casesensitive	<boolean>true</boolean>	this.casesensitive	<boolean>true</boolean>	
paramname	no value	paramname	no value	
parameters	<org.apache.tools.ant.typ...	parameters	<org.apache.tools.ant.type...	

Evaluation Dataset

- Our dataset contains 52 pairs of non-identical clones from 3 open source projects.
- Our dataset includes 38 Type II clones and 14 Type III clones, based on a well-known clone taxonomy [Roy et al.].

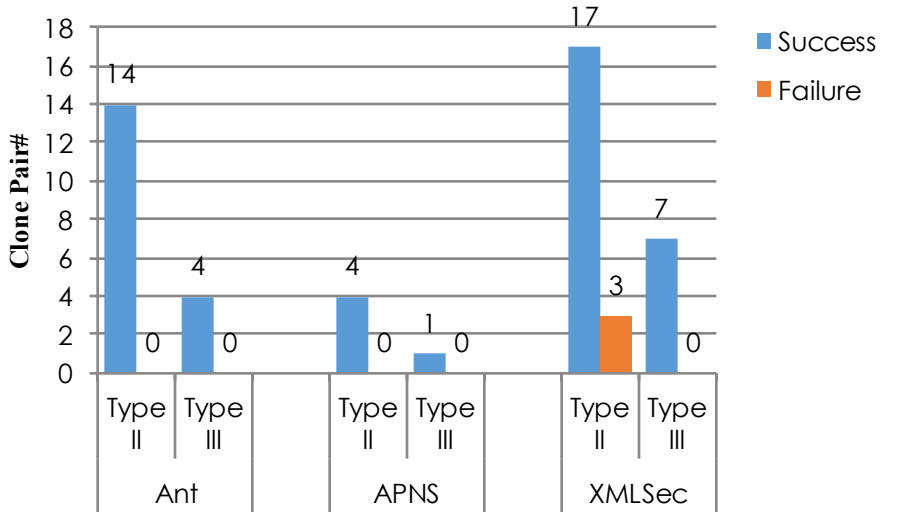
Subject	Version	Description	LOC	Test#	Branch	Stmt	Clone Pair
Apache Ant	1.9.6	A software build framework	267,048	1,864	45%	50%	18
Java APNS	1.0.0	A Java client for Apple Push Notification service (APNs)	8,362	103	59%	67%	7
XML Security	2.0.5	A XML signature and encryption library	121,594	396	59%	65%	27

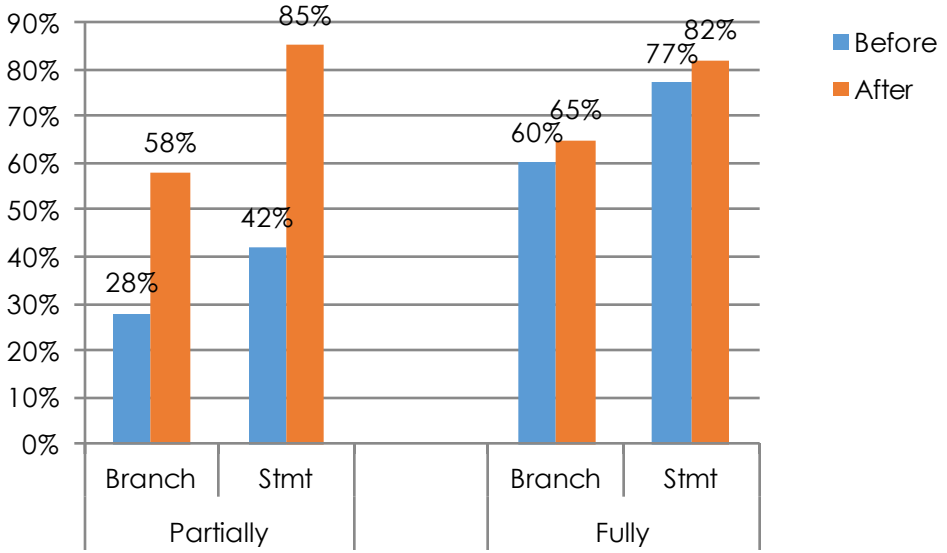
Research Questions

- RQ1. What is Grafter's transplantation capability?
- RQ2. How does Grafter compare with a static approach by Jiang et al. in its ability to detect differences in clones?
- RQ3. How sensitive is Grafter in detecting behavioral differences caused by mutants?

RQ1. Transplantation Success and Test Reuse Capability

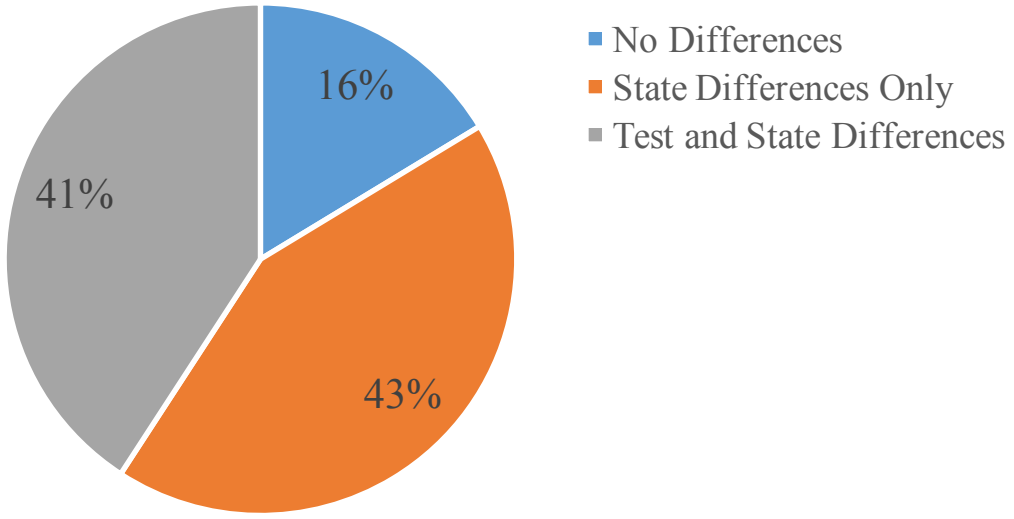
- Grafter successfully grafts 49 of 52 pairs of clones
- Grafter inserts 6 lines of stub code on average to ensure type safety
- Grafter doubles the test coverage for partially tested clone pairs



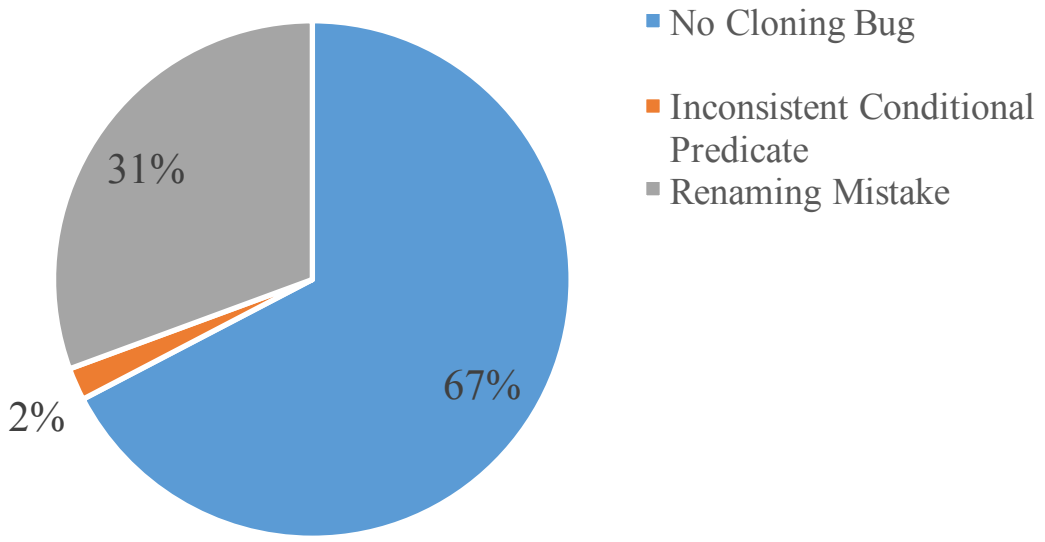


RQ2. Behavioral Difference Detection

- Jiang et al. [FSE'07] present a static approach that detects three pre-defined cloning inconsistencies: (1) renaming mistake, (2) control construct inconsistency, and (3) control predicate inconsistency
- Grafter exposes behavioral differences in 84% clone pairs while Jiang et al. detect syntactic inconsistency in 33% clone pairs only.



Grafter



Jiang et al.

RQ3. Robustness

- We systematically injected 361 mutants on 30 pairs of clones with no existing test behavioral differences
- We compare its behavioral differencing capability with a static approach by Jiang et al.

Operator	Description	Example
AOR	Arithmetic operator replacement	$a + b \rightarrow a - b$
LOR	Logical operator replacement	$a \wedge b \rightarrow a \mid b$
COR	Conditional operator replacement	$a \vee b \rightarrow a \,\&\&\, b$
ROR	Relational operator replacement	$a == b \rightarrow a >= b$
SOR	Shift operator replacement	$a \;?\; b \rightarrow a = b$
ORU	Operator replacement unary	$\neg a \rightarrow : a$
STD	Statement deletion operator: delete (omit) a single statement	$foo(a,b) \rightarrow // foo(a,b)$
LVR	Literal value replacement: replace by a positive value, a negative value or zero	$0 \rightarrow 1$

RQ3. Robustness

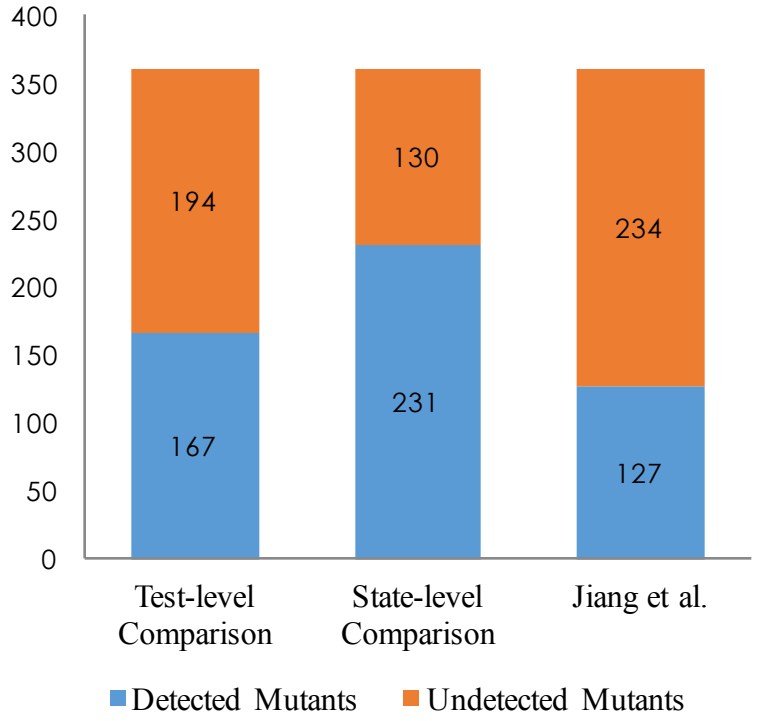
```
public void setType(String type) {
    if ( type != null  && type.length() == 0){
        this.type = null;
    } else {
        URI tmpType = null;
        try {
            tmpType = new URI(type);
        } catch (URISyntaxException ex) {
            …
        }
        this.type = tmpType.toString();
    }
}
```

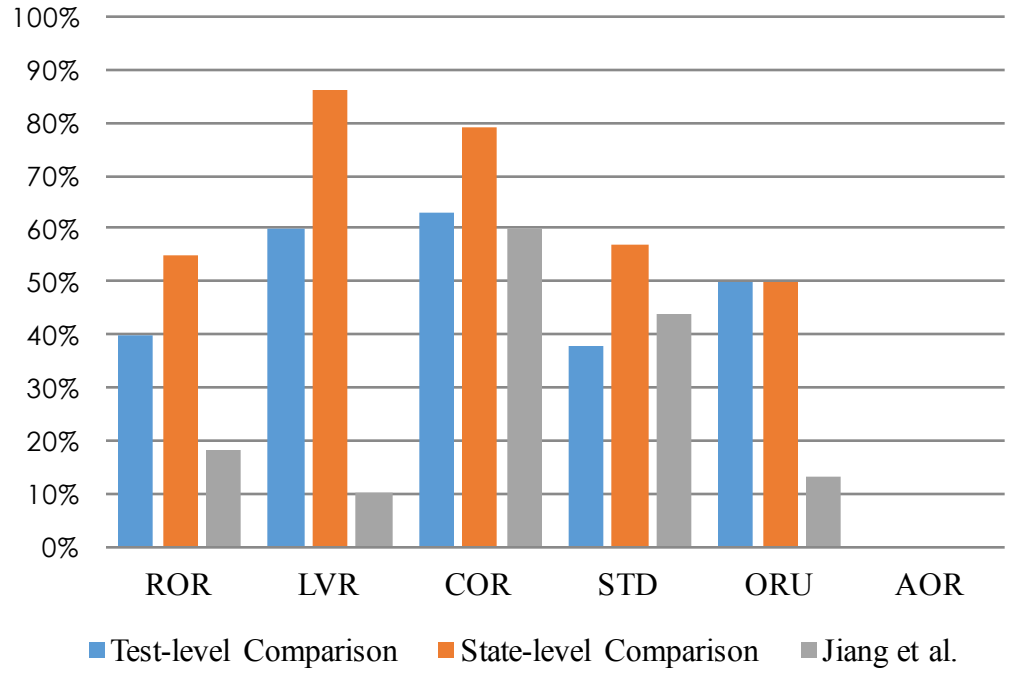
```
public void setEncoding(String encoding) {
    if (encoding == null && encoding.length() == 0){
        this.encoding = null;
    } else {
        URI tmpEncoding = null;
        try {
            tmpEncoding = new URI(encoding);
        } catch (URISyntaxException ex) {
            …
        }
        this.encoding = tmpEncoding.toString();
    }
}
```

Mutation Example from Apache XML Security

RQ3. Robustness

- Grafter detects 36% more mutants using the test-level comparison and almost 2X more mutants using the state-level comparison
- Grafter is less biased to mutant types than Jiang et al.





Conclusion

- This work introduces the first test transplantation and reuse approach for enabling runtime behavior comparison between clones.
- Grafter's code transplantation succeeds in 94% of the cases。
- The fine-grained differential testing can detect up to 2X more seeded faults than a baseline static cloning bug finder.

Q&A