

# Statistical-based Clone Detection

He Feng

Department of Physics

fenghe@vt.edu

Liuqing Li

Department of Computer Science

liuqing@vt.edu

# Outline

- Problem Definition
- Solution Design
- Implementation Status
- Technical Challenges
- Future Work

# Problem Definition

- Code Clone
  - Code fragment identical or similar to another
- Pros
  - Improves efficiency
  - Increases readability
- Cons
  - Low maintainability
  - Increases code size
- Clone Types (Type 1, 2, 3, 4)
- Techniques and Tools (Text, Token, Tree, Graph)

# Problem Definition

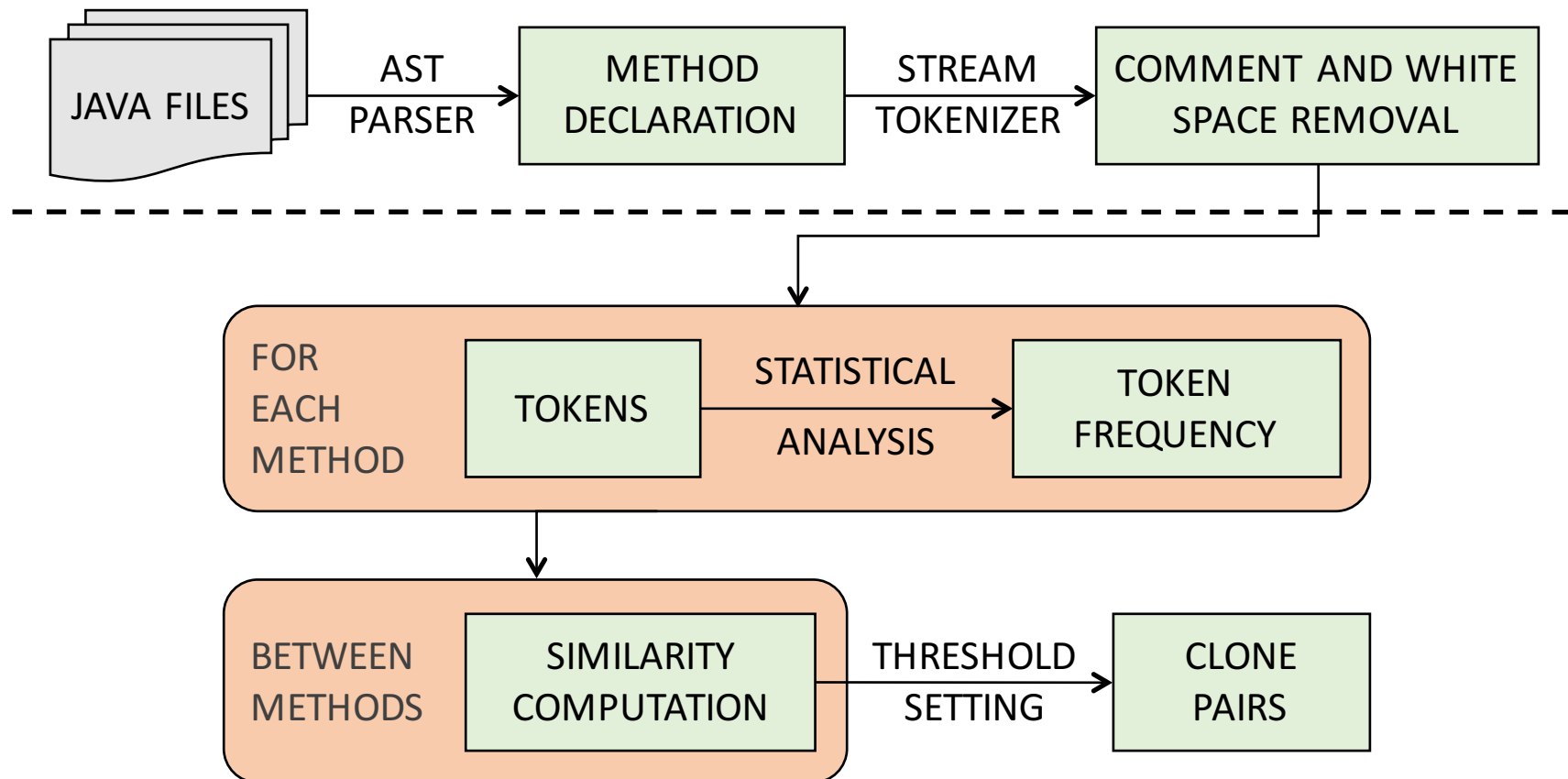
- Prerequisites
  - Token-based detection tool shows the best performance
  - Developers won't make dramatic changes in code clone
- Goal
  - Design and implement STCD (Statistical-based Clone Detection) tool to detect the Type 1, 2, 3 code clones between methods based on tokens
- Validity
  - Ambiguous match

# Solution Design

- Fragment A and B
  - Use a parser to catch all the tokens
  - Categorize key tokens into numbers, types, keywords, markers, operators, etc.
  - Accumulate the occurrences of each key token
  - Transform the fragment into a list of key token and frequency
  - Calculate the similarity between two lists
  - Set a threshold to the final output

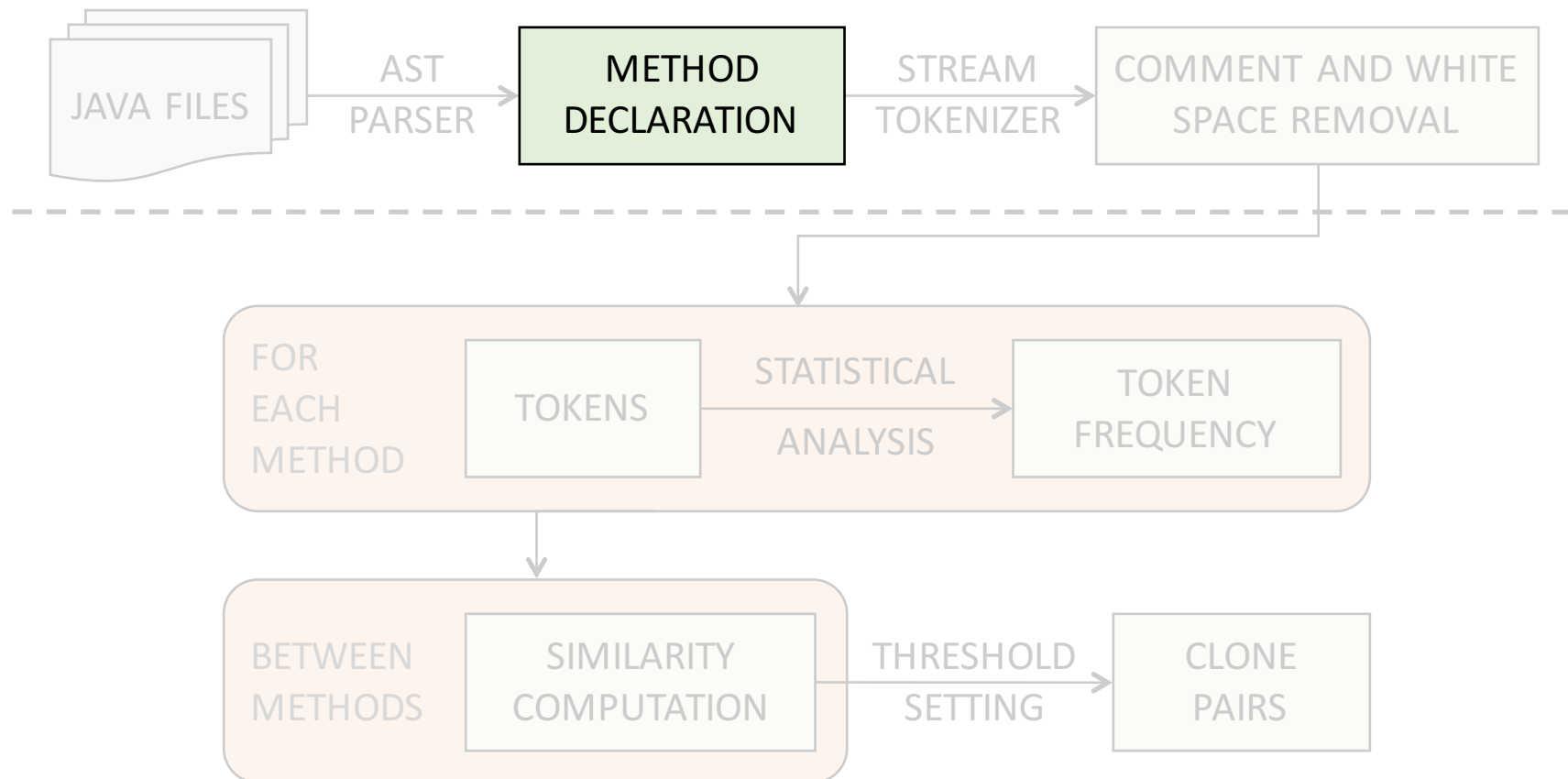
# Solution Design

- Overall Current Project Diagram



# Solution Design

- Overall Current Project Diagram



# Solution Design

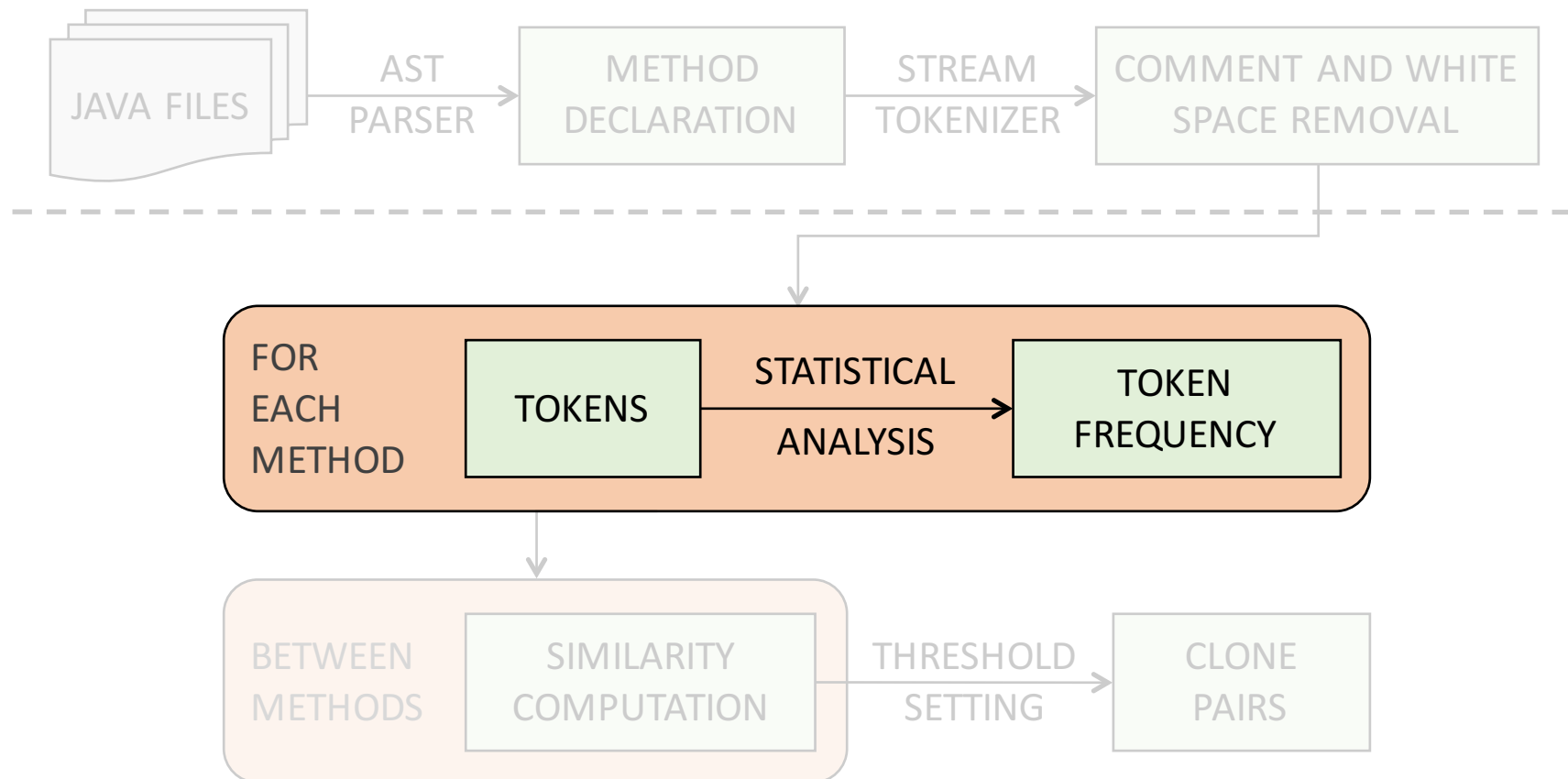
- Method Declaration
  - CompilationUnit `result`
  - MethodDeclaration `method`

Method Info.	Function
startLineNumber	result.getLineNumber(...)
endLineNumber	result.getLineNumber(...)
methodName	method.getName()
methodPara	method.parameters()
methodType	method.getReturnType2()
methodBody	method.getBody()



# Solution Design

- Overall Current Project Diagram



# Solution Design

- Token & Token Frequency
  - StreamTokenizer

Categories	Tokens
Number	123, 24.5, ...
Type	int, char, double, boolean, null, true, false, ...
Keyword	static, for, if, else, while, return, this, const, ...
Marker	{, }, [, ], (, )
Operator	+, -, *, /, %, ^, <, >, !, ...
Other1	variables, ...
Other2	?, #, \$, ...

# Solution Design

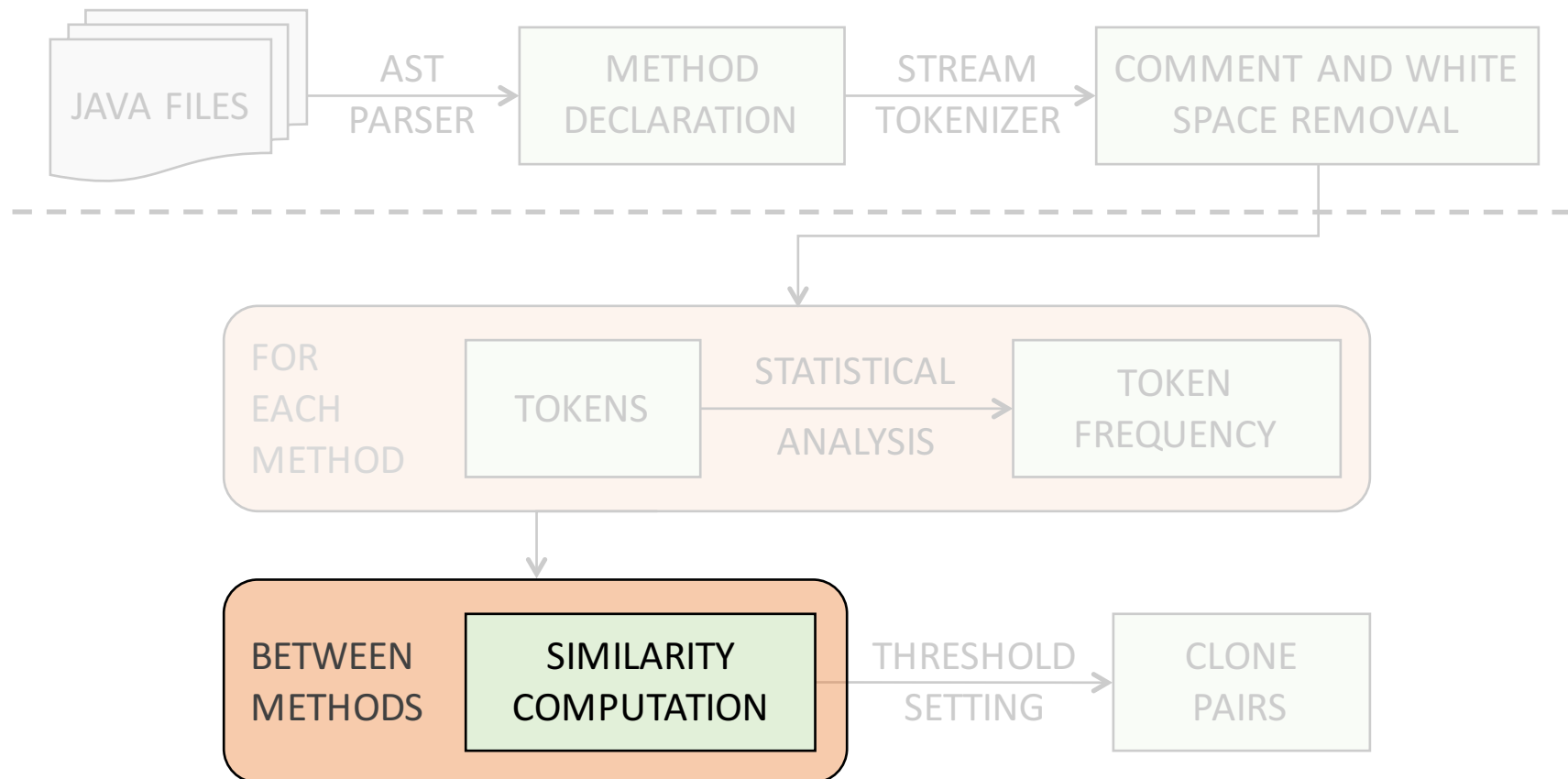
- Token & Token Frequency
  - tokenList: store token frequency
  - For two categories(e.g. Type and Keywords)

Type	Frequency
int	8
char	6
double	3
boolean	0
...	...
true	2
false	2

Keywords	Frequency
for	2
if	1
else	1
return	2
...	...
this	4
static	3

# Solution Design

- Overall Current Project Diagram



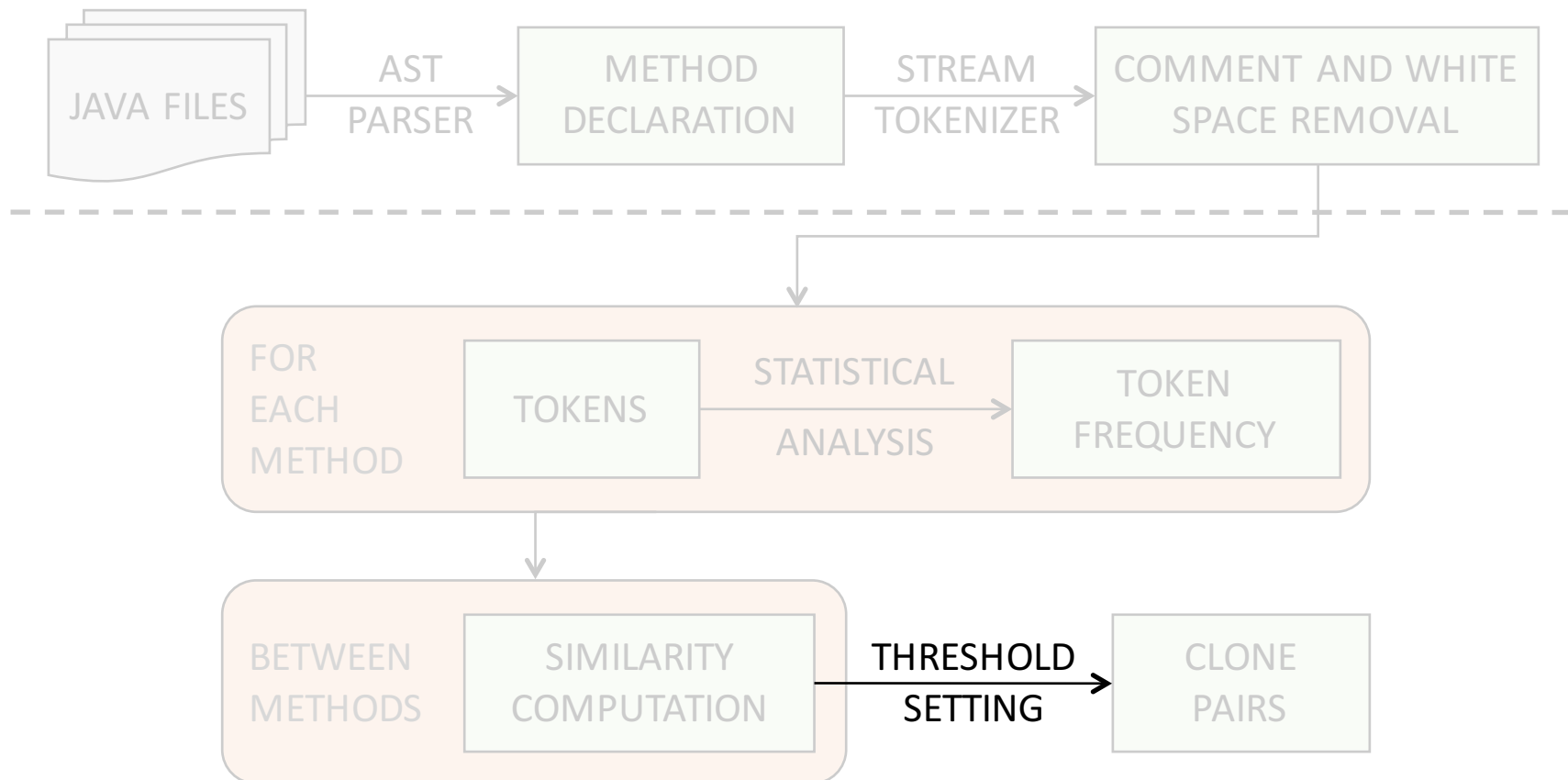
# Solution Design

- Similarity Computation
  - Method Similarity
  - 9-Dimensional Vector
    - $X = \langle X1, X2, X3, X4, X5, X6, X7, X8, X9 \rangle$
    - $Y = \langle Y1, Y2, Y3, Y4, Y5, Y6, Y7, Y8, Y9 \rangle$
    - $\text{Sim}(X1, Y1) = \text{bigram}(X1, Y1)$
    - $\text{Sim}(X2, Y2) = 1 / 0$
    - $\text{Sim}(X_{\text{else}}, Y_{\text{else}}) = 1 / 1 + \text{Distance}(X_{\text{else}}, Y_{\text{else}})$
    - $\text{Distance}(X_{\text{else}}, Y_{\text{else}})$ : Euclidean Distance
    - $\text{Sim}(X, Y) = \text{Sum}(w_i * \text{Sim}(X_i, Y_i)) \text{ (} i = 1, 2, \dots, 9 \text{)}$

Vector
methodPara
methodType
tokenListNum
tokenListType
tokenListKeyword
tokenListMarker
tokenListOperator
tokenListOther1
tokenListOther2

# Solution Design

- Overall Current Project Diagram



# Solution Design

- Threshold Setting 1
  - For Variables: DrawPointLine vs. PointLineDraw
  - tokenThreshold for bigram similarity
  - e.g. variableSimilarity > 0.7
- Threshold Setting 2
  - detectThreshold for method similarity
  - e.g. methodsSimilarity > 0.5
- Threshold Setting 3
  - lineThreshold for method lines
  - e.g. endLineNum - startLineNum > 7

# Implementation Status

- Source: <https://github.com/CSCC5704>
  - CodeClone.java
    - Detect cloned code in java files
  - MethodSimilarity.java
    - Calculate the similarity of two methods
  - BiGramSimilarity.java
    - Calculate the similarity of two strings by using bi-gram algorithm
  - ASTParserTool.java
    - Use JDT ASTParser to parse the java source code into methods
  - MethodTokenizerTool.java
    - Tokenize method body and get the frequency of tokens
  - MethodList.java, MethodVector.java
  - TokenList.java, TokenVector.java, ...



# Implementation Status

- TokenList.java

Token	Category	Freq
int	Type	1
for	Keyword	1
i	Other1	8
System	Other1	3
out	Other1	3
println	Other1	3
toBinary	Other1	1
Integer	Other1	1
toBinaryString	Other1	1
(	Marker	6

Token	Category	Freq
)	Marker	6
}	Marker	2
{	Marker	1
+	Operator	6
=	Operator	1
-	Operator	1
<	Operator	1
:	Other2	2
5.0	Num	1
33.0	Num	1

# Implementation Status

- Java Source Files from GitHub with LOC 200~3000
- Find out pairs of clone code
  - Output Format
    - Clone Group # --> Similarity: #
    - Method Name 1    Start Line #    End Line #
    - Method Name 2    Start Line #    End Line #
- Matches well with manual examination

# Implementation Status

```
86     public void testUIComputation() throws Exception {
87         // populate text fields with values
88         populate();
89         // trigger computation by using the button
90         WebElement button = driver.findElement(By.className("UIAButton"));
91         button.click();
92         // is sum equal ?
93         WebElement texts = driver.findElement(By.className("UIAStaticText"));
94         assertEquals(String.valueOf(values.get(0) + values.get(1)), texts.getText());
95     }

142     public void testFindElementByClassName() throws Exception {
143         Random random = new Random();
144
145         WebElement text = driver.findElementByClassName("UIATextField");
146         int number = random.nextInt(MAXIMUM - MINIMUM + 1) + MINIMUM;
147         text.sendKeys(String.valueOf(number));
148
149         driver.findElementByClassName("UIAButton").click();
150
151         // is sum equal ?
152         WebElement sumLabel = driver.findElementByClassName("UIAStaticText");
153         assertEquals(String.valueOf(number), sumLabel.getText());
154     }
```

Source: <https://github.com/appium/sample-code>  
Clone Group 3 --> Similarity: 0.5102

# Implementation Status

```
142     public void testFindElementByClassName() throws Exception {
143         Random random = new Random();
144
145         WebElement text = driver.findElementByClassName("UITextField");
146         int number = random.nextInt(MAXIMUM - MINIMUM + 1) + MINIMUM;
147         text.sendKeys(String.valueOf(number));
148
149         driver.findElementByClassName("UIAButton").click();
150
151         // is sum equal ?
152         WebElement sumLabel = driver.findElementByClassName("UIAStaticText");
153         assertEquals(String.valueOf(number), sumLabel.getText());
154     }
155
156     public void testFindElementsByClassName() throws Exception {
157         Random random = new Random();
158
159         WebElement text = driver.findElementsByClassName("UITextField").get(1);
160         int number = random.nextInt(MAXIMUM - MINIMUM + 1) + MINIMUM;
161         text.sendKeys(String.valueOf(number));
162
163         driver.findElementByClassName("UIAButton").click();
164
165         // is sum equal ?
166         WebElement sumLabel = driver.findElementsByClassName("UIAStaticText").get(0);
167         assertEquals(String.valueOf(number), sumLabel.getText());
168     }
169 }
```

Source: <https://github.com/appium/sample-code>

Clone Group 5 --> Similarity: 0.6676

# Implementation Status

```
142     public void testFindElementByClassName() throws Exception {
143         Random random = new Random();
144
145         WebElement text = driver.findElementByClassName("UITextField");
146         int number = random.nextInt(MAXIMUM - MINIMUM + 1) + MINIMUM;
147         text.sendKeys(String.valueOf(number));
148
149         driver.findElementByClassName("UIAButton").click();
150
151         // is sum equal ?
152         WebElement sumLabel = driver.findElementByClassName("UIAStaticText");
153         assertEquals(String.valueOf(number), sumLabel.getText());
154     }
155
156
157
158
159
160
161
162
163
164
165
166
167
168
169
170
171
172     public void testAttribute() throws Exception {
173         Random random = new Random();
174
175         WebElement text = driver.findElement(By.xpath("//UITextField[1]"));
176
177         int number = random.nextInt(MAXIMUM - MINIMUM + 1) + MINIMUM;
178         text.sendKeys(String.valueOf(number));
179
180         assertEquals("TextField1", text.getAttribute("name"));
181         assertEquals("TextField1", text.getAttribute("label"));
182         assertEquals(String.valueOf(number), text.getAttribute("value"));
183     }
```

Source: <https://github.com/appium/sample-code>  
Clone Group 6 --> Similarity: 0.6190

# Implementation Status

- Apache Commons
  - <http://commons.apache.org>
- Two Files
  - FileUtils.java
    - Total methods: 106
    - Cloned methods: 11 pairs (no clusters)
    - Ratio: 20.8%
  - IOUtils.java
    - Total methods: 105
    - Cloned methods: 4 pairs (no clusters)
    - Ratio: 7.62%

# Technical Challenges

- ASTParser
  - Excessive time cost
  - Improve the method parsing process
  - e.g. Regular expression
- Manual weights and threshold
  - 9-Dimentional ( $w_1, \dots, w_9$ )
  - Machine Learning Techniques (Training Data)
  - e.g. Multilayer perceptron (MLP)

# Future Work

- Variables Comparison
  - Bigram Algorithm
- Data Collection
  - Training Data for Machine Learning
  - Test Data for results comparison
- Results Comparison
  - Precision & Recall
- UI Development
  - Java WindowBuilder



**Thank you !**