Statistical-based Clone Detection (STCD)

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 – Type 4
- Techniques and Tools
  - Text-based, token-based, tree-based, graph-based

Our Proposed Solution

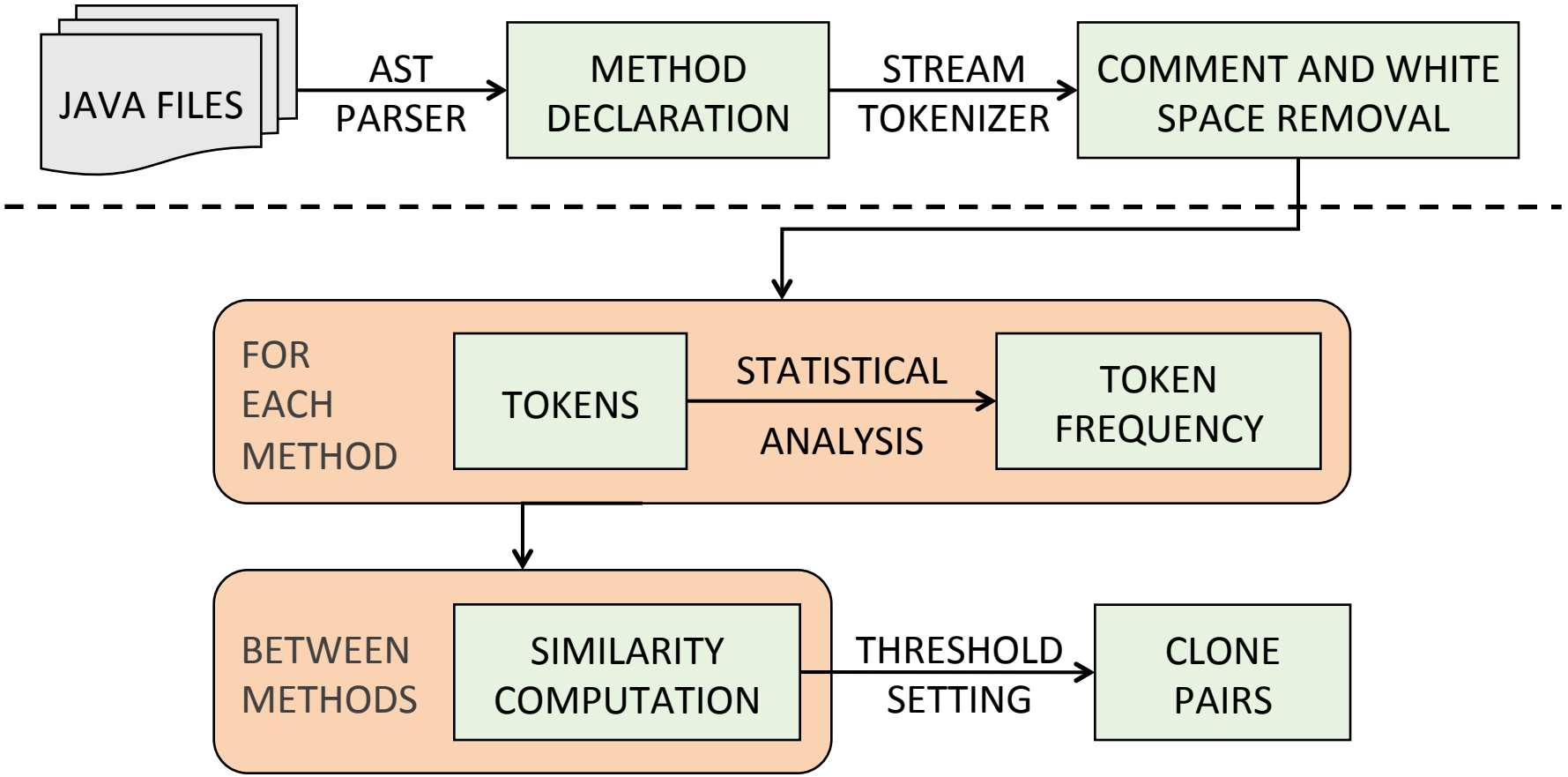
- Prerequisites
  - Token-based detection tool shows the best performance
  - Developers won't make dramatic changes in code clone
- Goal
  - Design and implement a Statistical-based Clone Detection(STCD) tool to detect the Type 1, 2, 3 code clones between code methods based on tokens
- Validity
  -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

Our Proposed Solution

- Fragment A and B
  - Use a parser to catch all the tokens
  - Categorize key tokens into types, variables, identifiers, 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

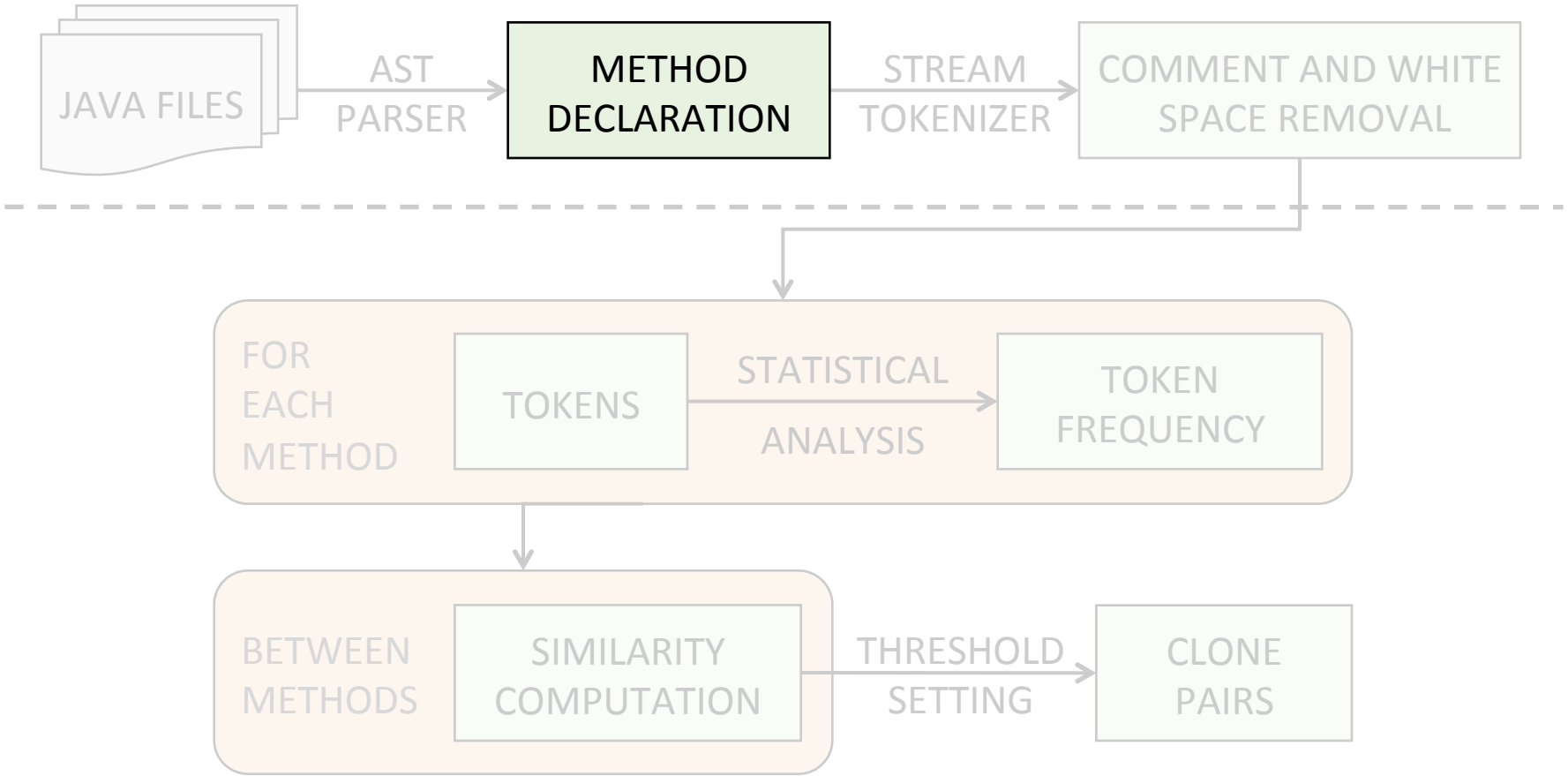
Structure of Designed Program

- Overall Current Project Diagram



Structure of Designed Program

- Overall Current Project Diagram



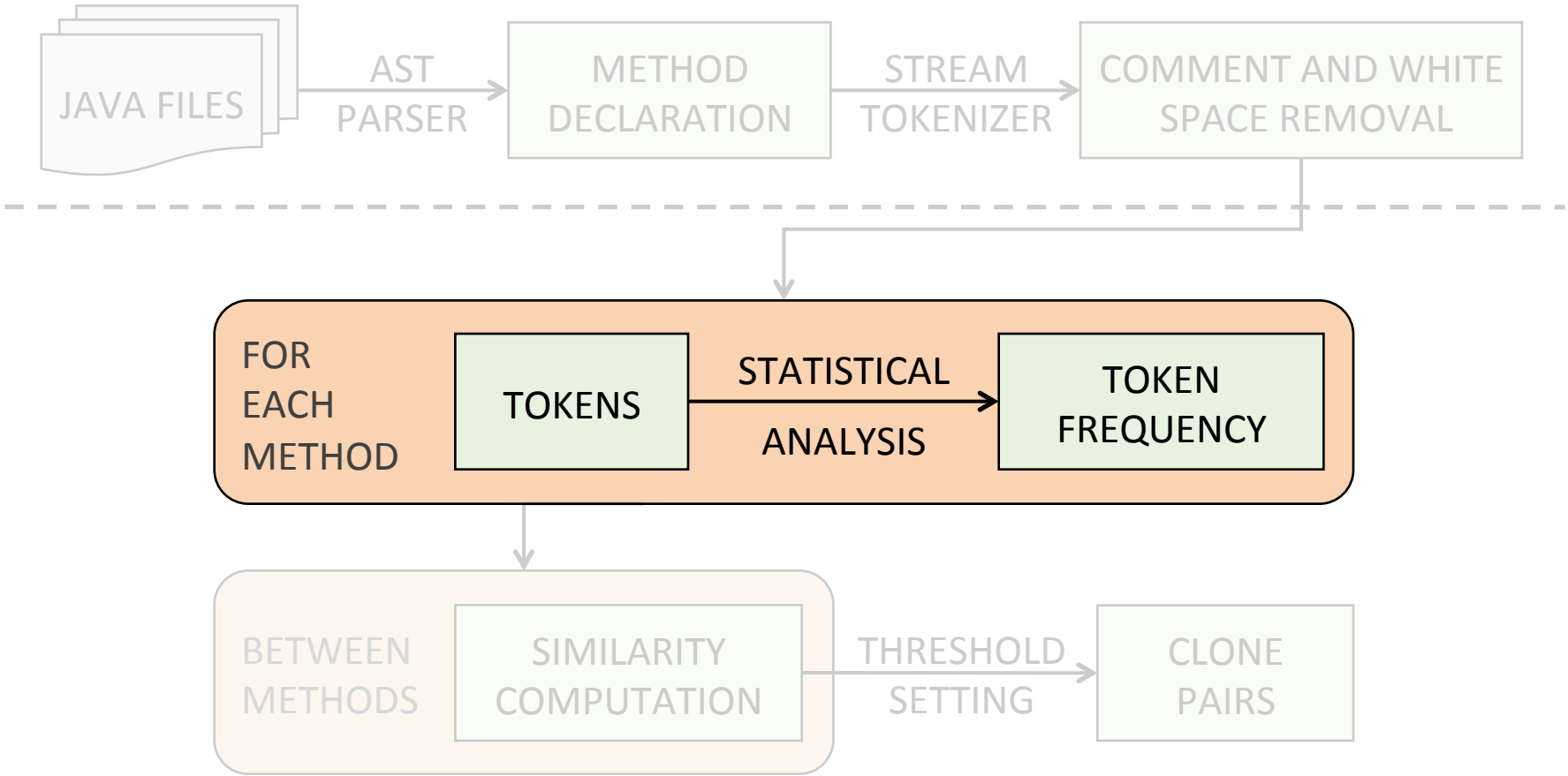
Structure of Designed Program

- 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()

Structure of Designed Program

- Overall Current Project Diagram



Structure of Designed Program

- 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	?, #, $, …

Structure of Designed Program

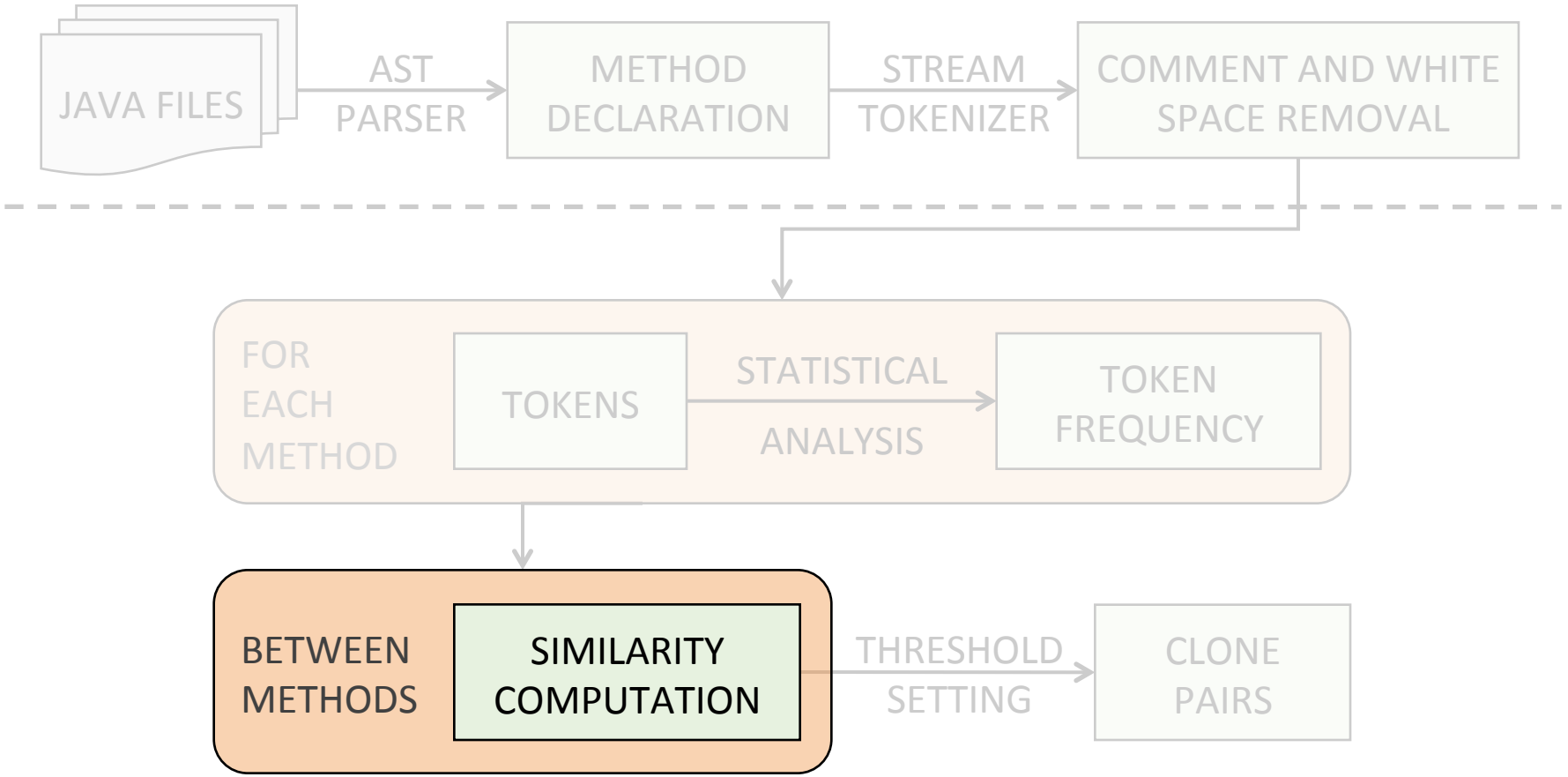
- Token & Token Frequency
  - tokenList: store token frequency
  - For two categories(e.g. Type and Keywords)

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

Structure of Designed Program

- Overall Current Project Diagram



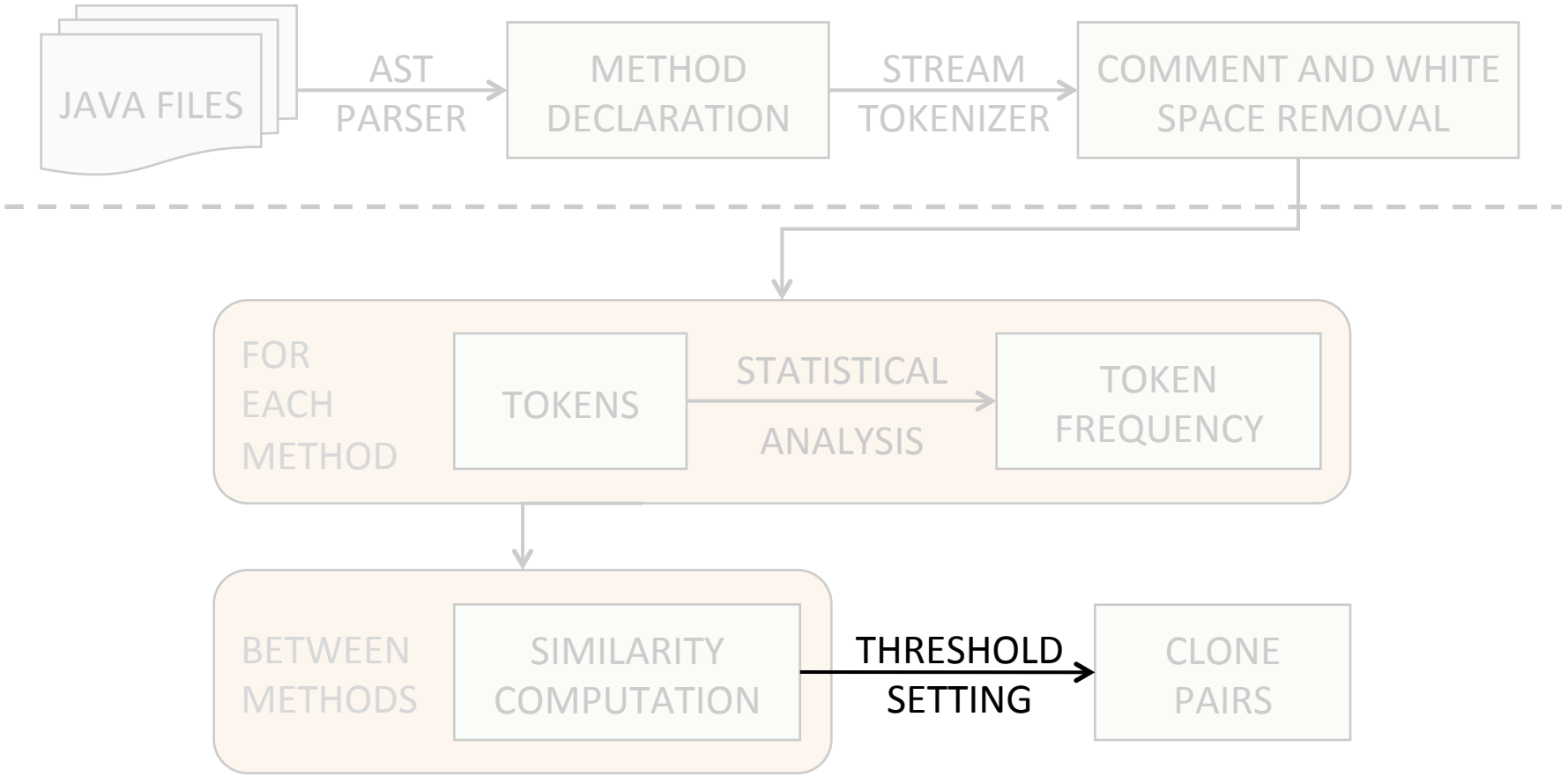
Structure of Designed Program

- Similarity Computation
  - Method Similarity
  - 9-Dimentional Vetcor
    - X = <X1, X2, X3, X4, X5, X6, X7, X8, X9>
    - Y = <Y1, Y2, Y3, Y4, Y5, Y6, Y7, Y8, Y9>
    - Sim(X1, Y1) = bigram(X1, Y1)
    - Sim(X2, Y2) = 1 / 0
    - Sim(X_{else}, Y_{else}) =
    - Sim(X, Y) = Sum(wi * Sim(Xi, Yi)) (i = 1, 2, …, 9)

Vector
methodPara
methodType
tokenListNum
tokenListType
tokenListKeyword
tokenListMarker,
tokenListOperator
tokenListOther1
tokenListOther2

Structure of Designed Program

- Overall Current Project Diagram



Structure of Designed Program

- Threshold Setting
  - tokenListOther1 (Variables)
    - Problem: DrawPointLine vs. PointLineDraw
    - Threshold 1: bigram similarity
    - e.g. tokenThreshold > 0.7
  - Method
    - Threshold 2: method similarity
    - e.g. detectThreshold > 0.5
    - Threshold 3: method lines
    - e.g. endLineNum - startLineNum > 7

Implementation Status

- Source code: https://github.com/CSCC5704
- CodeClone.java - the main function to detect cloned code in java files
- ASTParserTool.java - use JDT ASTParser to parse the jave source code into methods
- MethodTokenizerTool.java - tokenize the method body and get the frequency of selected tokens
- BiGramSimilarity.java - calculate the similarity of two strings by using the bi-gram algorithm
- MethodSimilarity.java - calculate the similarities of two methods (e.g. methodPara, methodType)
- MethodList.java - a list of MethodVector and some local functions

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
)	Marker	6

}	Marker	2
{	Marker	1
+	Operator	6
=	Operator	1
-	Operator	1
<	Operator	1
.	Other2	7
:	Other2	2
5.0	Num	1
33.0	Num	1

Test Result

- Java source files from GitHub with LOC 200~3000

- Find out clone fragments
  - Output format:
    - Clone Group # → Similarity: #
    - Method Name 1 Start Line # End Line #
    - Method Name 2 Start Line # End Line #

- Matches well with manual examination

Test Result Example 1

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

```
85      @Test
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      }
141     @Test
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     }
```

```
Clone Group 3 --> Similarity :0.5102
testUIComputation              85              94
testFindElementByClassName     141             153
```

Test Result Example 2

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

```
141    @Test
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    }
171    @Test
172    public void testAttribute() throws Exception {
173        Random random = new Random();
174
175        WebElement text = driver.findElement(By.xpath("//UIATextField[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    }
```

```
Clone Group 6 --> Similarity :0.6190
testFindElementByClassName          141          153
  testAttribute                     171          182
```

Test Result Example 3

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

```
141    @Test
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    }
156    @Test
157    public void testFindElementsByClassName() throws Exception {
158      Random random = new Random();
159
160      WebElement text = driver.findElementsByClassName("UIATextField").get(1);
161      int number = random.nextInt(MAXIMUM - MINIMUM + 1) + MINIMUM;
162      text.sendKeys(String.valueOf(number));
163
164      driver.findElementByClassName("UIAButton").click();
165
166      // is sum equal ?
167      WebElement sumLabel = driver.findElementsByClassName("UIAStaticText").get(0);
168      assertEquals(String.valueOf(number), sumLabel.getText());
169    }
```

```
Clone Group 5 --> Similarity :0.6676
testFindElementByClassName          141          153
testFindElementsByClassName          156          168
```

- Apache Commons
- http://commons.apache.org
- FileUtils.java total methods: 106
  11 pairs of clones ratio: 20.8%
- IOUtils.java 105 4 pairs ratio: 7.62%

Challenges

- Use ASTParser tool to catch the tokens, excessive time cost
  - Improve the tokenize process, reduce time cost
  - e.g regular expression
- Manually set weight and threshold, need to be improved
  - Use machine learning to set weights
  - e.g Multilayer perceptrons (MLP)

Future Work

- Variables comparison
- Data collection
- Results comparison
- UI development

Thank you !