# 추상 구문 트리에 기반한 코드 변화 분석

## AN ANALYSIS OF CODE CHANGE BASED ON EDIT SCRIPT OF ABSTRACT SYNTAX TREE


이 창 공, 나 예 원, 최 윤 호

한동대학교 전산전자공학부

이 건 우, 최 명 석

한국과학기술정보연구원 기계학습데이터연구단

남 재 창

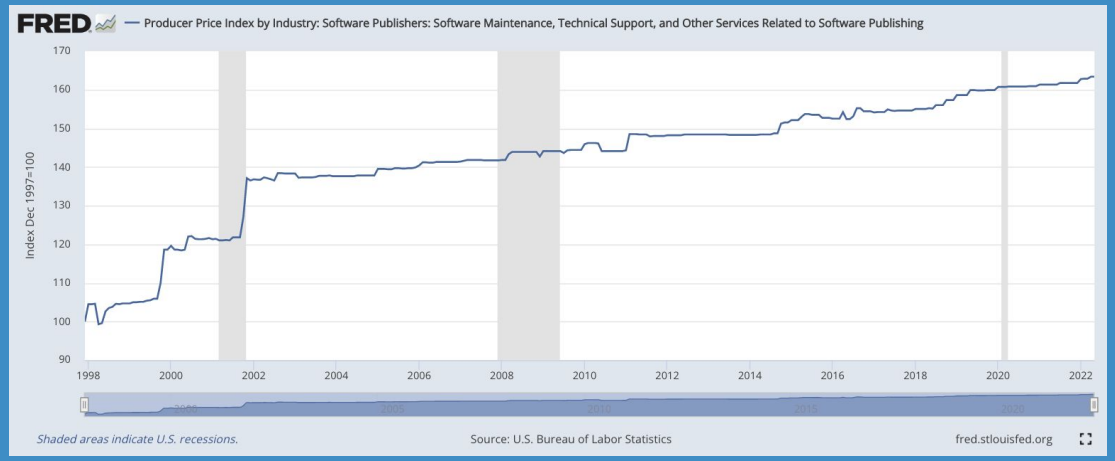
한동대학교 전산전자공학부

# PROBLEM

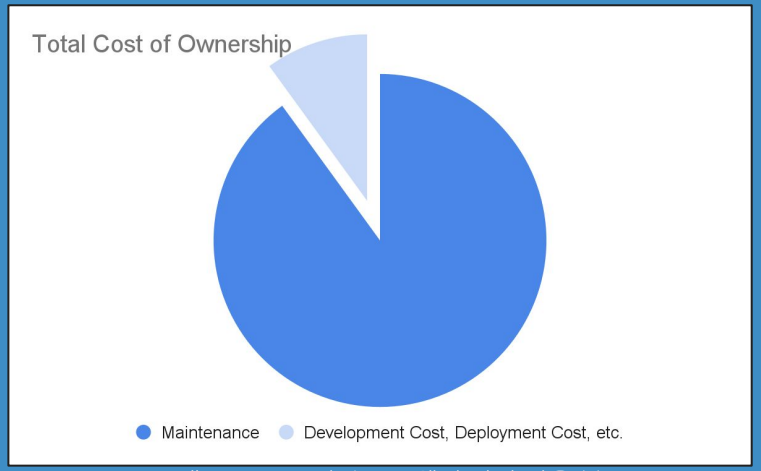
# PROBLEM

## TO ERR IS MAN

# PROBLEM




그래프 1. 연도별 소프트웨어 관련 비용 (미 노동 통계국)




그래프 2. 2022년 소프트웨어 관련 비용 분포

U.S. Bureau of Labor Statistics, Producer Price Index by Industry: Software Publishers: Software Maintenance, Technical Support, and Other Services Related to Software Publishing [PCU511210511210504], retrieved from FRED, Federal Reserve Bank of St. Louis; https://fred.stlouisfed.org/series/PCU511210511210504, February 5, 2023.

Andreev, Vladislav. “Software Maintenance Costs: Factors & Ways to Reduce.” *Custom Software Development Company*, 28 Feb. 2022, https://maddevs.io/customer-university/software-maintenance-costs/#software-maintenance-vs-development-cost.

# PROBLEM

Software Maintenance

Debugging

Bug Detection

Bug Reproduction

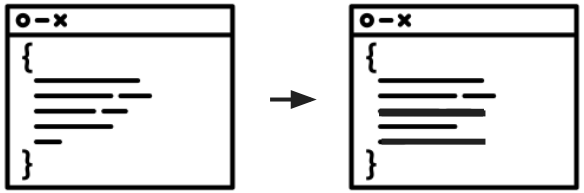
Testing

Bug Fix

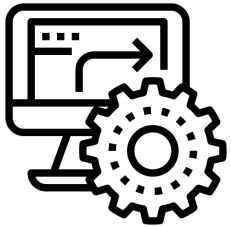
**> $31.2M**

# PROBLEM

Debugging



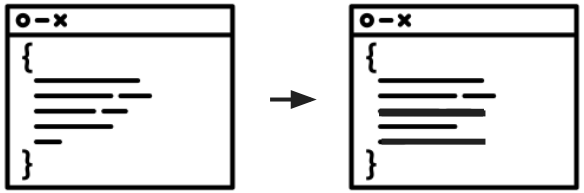
Change



Automation

# PROBLEM

Debugging



Change

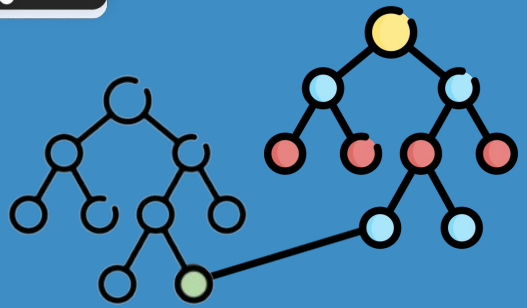
Case 1:
Change A to B
Case 2:
Change C to B
Case 3:
Change Y to V
....

Pattern/Rule

T. Gregory, “The Economic Impacts of Inadequate Infrastructure for Software Testing.” technical report 02-3, NIST, May 2002”

# BACKGROUND

# BACKGROUND

Gabel, Mark, and Zhendong Su. "A study of the uniqueness of source code." Proceedings of the eighteenth ACM SIGSOFT international symposium on Foundations of software engineering. 2010.
Hindle, Abram, et al. "On the naturalness of software." Communications of the ACM 59.5 (2016): 122-131.
Mockus, Audris. "Large-scale code reuse in open source software." First International Workshop on Emerging Trends in FLOSS Research and Development (FLOSS'07: ICSE Workshops 2007). IEEE, 2007.
Mockus, Audris. "Amassing and indexing a large sample of version control systems: Towards the census of public source code history." 2009 6th IEEE International Working Conference on Mining Software Repositories. IEEE, 2009.

# BACKGROUND

## Definition. Repeatedness

Let, **s** and **t** be subtrees of ASTs from two different stages of a program evolution for program **P**.

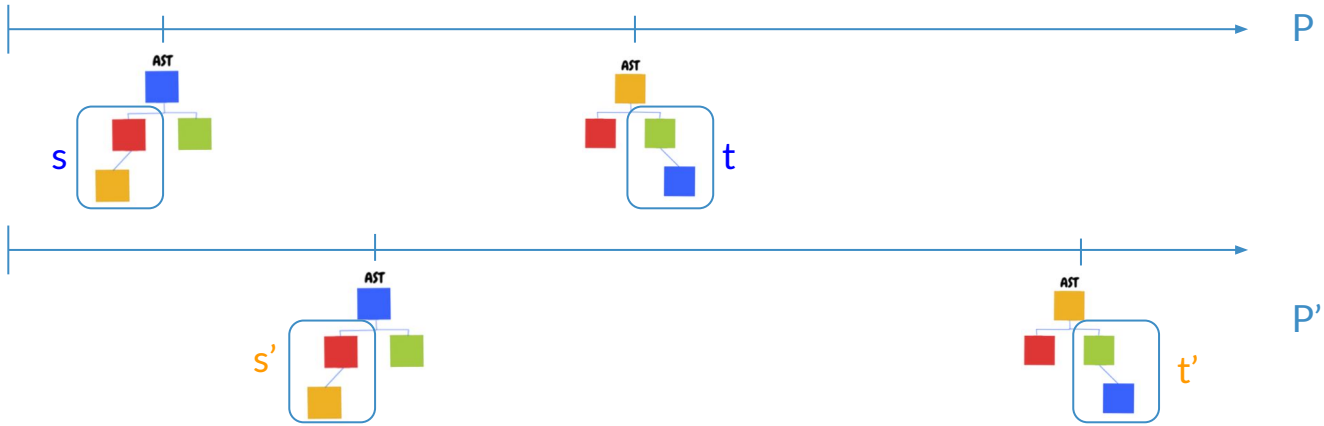
If **(s, t)** is a change representation of the **s subtree** to the **t subtree** and there exists **(s', t')** from another program **P'** such that **s and s'**, and **t and t'** have **equivalent abstract syntactic structure**,

Then, **(s, t) and (s', t') have a repeated pattern**.

Nguyen, Hoan Anh, et al. "A study of repetitiveness of code changes in software evolution." 2013 28th IEEE/ACM International Conference on Automated Software Engineering (ASE). IEEE, 2013.

# BACKGROUND

## Definition. Repeatedness




$$pattern(s, t) \approx pattern(s', t')$$

Nguyen, Hoan Anh, et al. "A study of repetitiveness of code changes in software evolution." 2013 28th IEEE/ACM International Conference on Automated Software Engineering (ASE). IEEE, 2013.

# BACKGROUND

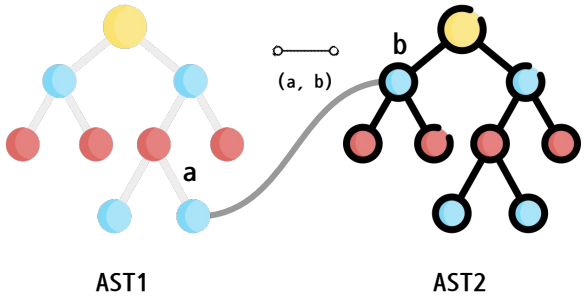
## GUMTREE

### CHANGE

```
@@ -2,7 +2,6 @@

 public class Main {
     public static void main(String[] args) {
-        System.out.println("Hello World!");
         String isHellow = "Hello";
         if(isHellow.equals("Hello")){
             System.out.println("Hello");
```

### MAPPING




### RESULT

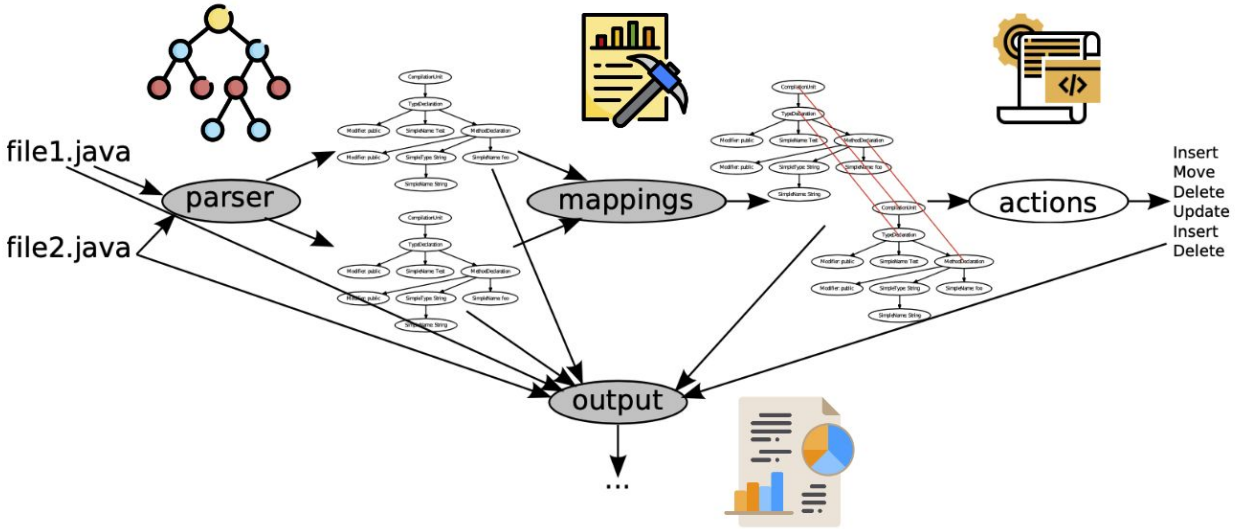
```
===
delete-tree
---
ExpressionStatement [88,123]
MethodInvocation [88,122]
METHOD_INVOCATION_RECEIVER [88,98]
QualifiedName: System.out [88,98]
SimpleName: println [99,106]
METHOD_INVOCATION_ARGUMENTS [107,121]
StringLiteral: "Hello World!" [107,121]
```

Falleri, J. R., Morandat, F., Blanc, X., Martinez, M., and Monperrus, M., "Fine-grained and accurate source code differencing," Proceedings of the 29th ACM/IEEE international conference on Automated software engineering, pp. 313-324, 2014.

# BACKGROUND

## GUMTREE

### MAPPING



# BACKGROUND

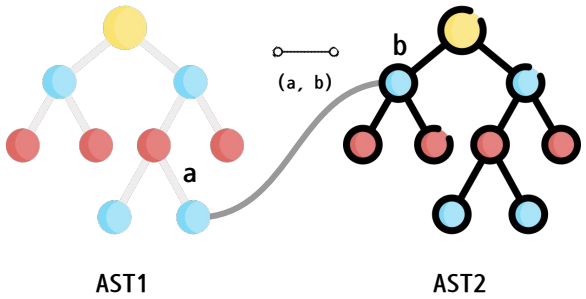
## GUMTREE

### CHANGE

```
@@ -2,7 +2,6 @@

 public class Main {
     public static void main(String[] args) {
-        System.out.println("Hello World!");
         String isHellow = "Hello";
         if(isHellow.equals("Hello")){
             System.out.println("Hello");
```

### MAPPING




### RESULT

```
===
delete-tree
---
ExpressionStatement [88,123]
MethodInvocation [88,122]
METHOD_INVOCATION_RECEIVER [88,98]
QualifiedName: System.out [88,98]
SimpleName: println [99,106]
METHOD_INVOCATION_ARGUMENTS [107,121]
StringLiteral: "Hello World!" [107,121]
```

# BACKGROUND

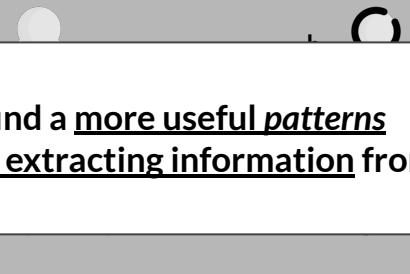
## GUMTREE

### CHANGE

```
@@ -2,7 +2,6 @@

 public class Main {
     public static void main(String[
-        System.out.println("Hello W
         String isHellow = "Hello";
         if(isHellow.equals("Hello")){
             System.out.println("Hello "),
```

### MAPPING




### RESULT

```
===
                                                    23]
                                                    RECEIVER [88,98]
QualifiedName: System.out [88,98]
SimpleName: println [99,106]
METHOD_INVOCATION_ARGUMENTS [107,121]
StringLiteral: "Hello World!" [107,121]
```

**RQ. Can we find a more useful *patterns* by selectively extracting information from Gumtree algorithm?**

# APPROACH

# APPROACH

**Step 1: Collecting Commits**

**Step 2: AST Representation &**
**Extracting Gumtree information to make formatted patterns.**

**Step 3: Encoding & Grouping patterns**

# APPROACH

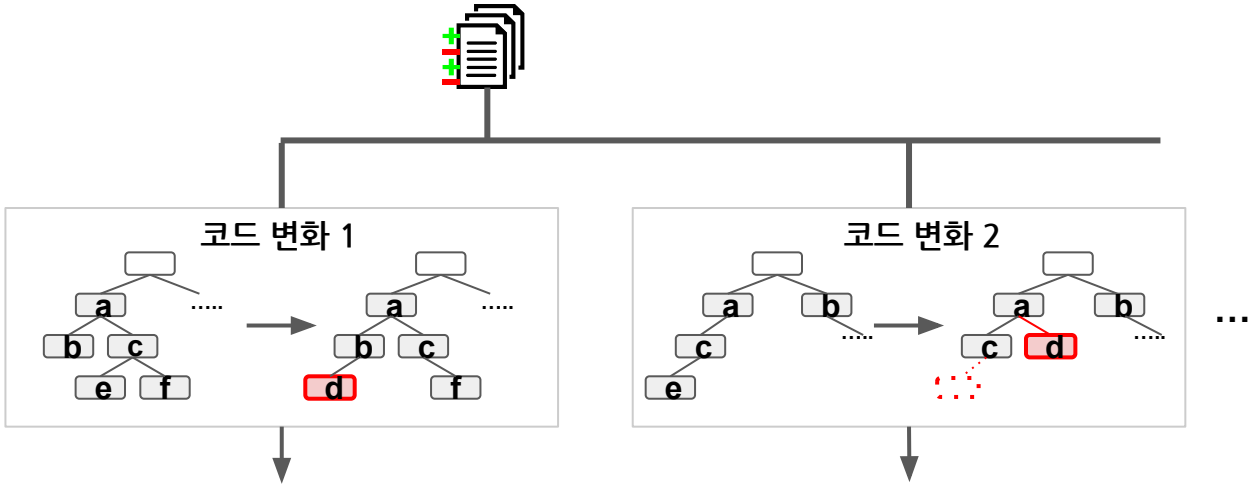
## Step 1: Collecting Commits


| 언어 | Java | Python | C |
|---|---|---|---|
| 총 프로젝트 수 | 219 | 25 | 16 |
| 총 커밋 수 | 1,219,135 | 327,222 | 49,757 |
| 총 코드 변화 수 | 17,589,637 | 2,123,880 | 108,807 |

**Total Project #: 260**
**Total Commit #: 1,596,114**
**Total Change #: 19,822,324**

# APPROACH

**Step 2: AST Representation &**
**Extracting Gumtree information to make formatted patterns.**




*Pattern* = “**Insert d at b**” + “**Delete e from c, Insert d at a**” + …

# APPROACH

**Step 2: AST Representation &**
**Extracting Gumtree information to make formatted patterns.**

*Pattern* = **“Insert d at b” + “Delete e from c, Insert d at a” + …**

Change Type
ex) insert, delete, move, update

Root node of the subtree
ex) MethodInvocation, VarDeclaration

Parent node of the subtree
ex) WhileStatement, IfStatement

# APPROACH

**Step 3: Encoding & Grouping patterns**

*Pattern* → 7fb22118c0d89133e337bc87f333d883bee3cec479cb9135a5f0082f8f3f3bc0

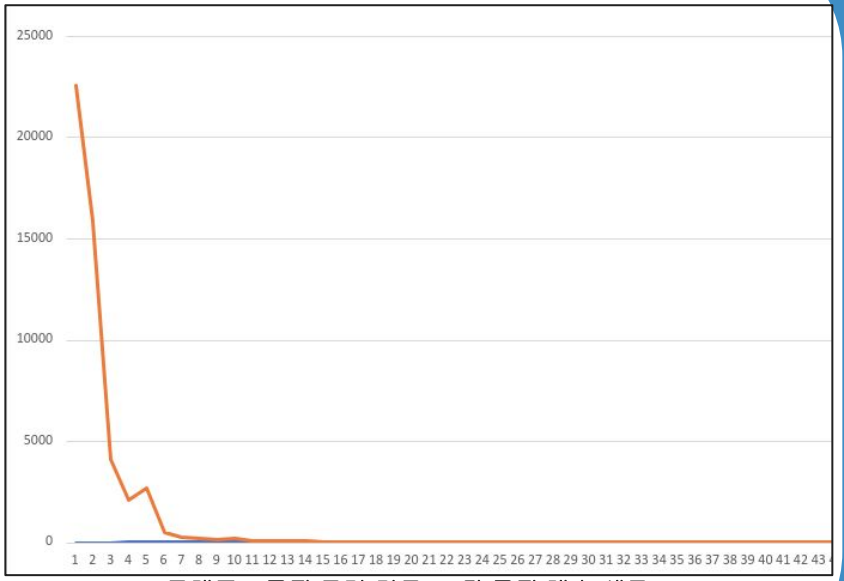
# EVALUATION

# EVALUATION

## a. Cluster Analysis

| 언어 | Java | Python | C |
|---|---|---|---|
| 총 프로젝트 수 | 219 | 25 | 16 |
| 총 커밋 수 | 1,219,135 | 327,222 | 49,757 |
| 총 코드 변화 수 | 17,589,637 | 2,123,880 | 108,807 |
| 군집 크기가 2개 이상인 코드 변화의 수 | 16,859,221 (95.8%) | 2,004,352 (94.4%) | 94,593 (87%) |
| 군집 수 | 1,156,756 | 196,374 | 21,996 |
| 최대 군집 크기 | 1,031,914 | 112,642 | 4,298 |
| 평균 군집 크기 | 6 | 4 | 3 |

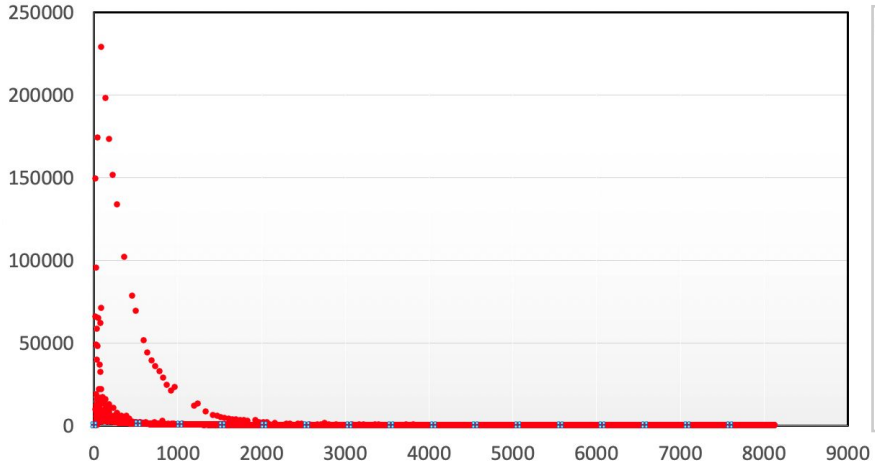
표 1. 코드 변화 군집화 결과 통계



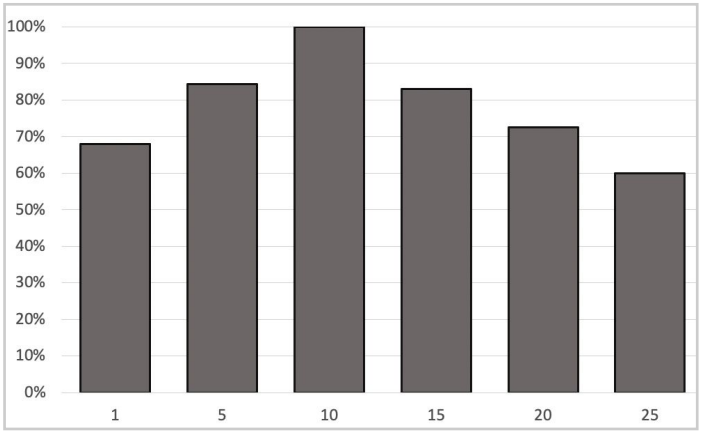

그래프 3.군집 크기(가로, X)당 군집 개수(세로, Y)

# EVALUATION

## b. Script-length Analysis




그래프 4. 변경 스크립트의 길이(가로)와 군집 크기(세로)의 관계 그래프




그래프 5. 변경 스크립트의 길이에 따른 코드 변화의 유사도

# EVALUATION

## c. Examples

***keras/wrappers/scikit_learn.py***

```
        import copy
        import inspect
        import types
+       import numpy as np
        from ..utils.np_utils import to_categorical
        from ..models import Sequential
```

***streamlit/tiny_notebook/protobuf/__init__.py***

```
        from Div_pb2 import Div
        from Element_pb2 import Element
        from Delta_pb2 import Delta, DeltaList
+       from DataFrame_pb2 import DataFrame, AnyArray, Table

        # Clear out all temporary variables.
        sys.path.pop()
```

***transformers/hubconf.py b/hubconf.py***

```
                ...
        gpt2LMHeadModel, gpt2DoubleHeadsModel )
+       from hubconfs.transformer_xl_hubconf import (
+       transformerXLTokenizer,
+       transformerXLModel,
+       transformerXLLMHeadModel
+       )
```

**Script: InsertTree-SimpleStmt@FileInput**

**Group Size: 35**

**Script Length: 1**

**Relavant(Similar) Changes: 14(40%)**

# EVALUATION

## c. Examples

***apache/axis-axis2-java-core/modules/kernel/src/org/apache/axis2/deployment/DeploymentEngine.java***

```
+       protected Scheduler scheduler;
        ...
        protected void startSearch(RepositoryListener listener)
-       Scheduler scheduler = new Scheduler();
+       scheduler = new Scheduler();
        ...
+       public void cleanup() {
+           ...
+       }
```

***apache/ant-ivyde/org.apache.ivyde.eclipse/src/java/org/apache/ivyde/eclipse/ui/ConfTableViewer.java***

```
+       private Link select;
        public ConfTableViewer(Composite parent, int style) {
            ...
-           Link select = new Link(this, SWT.PUSH);
+           select = new Link(this, SWT.PUSH);
        ...
+       public void setEnabled(Boolean enabled) {
+           ...
+       }
```

**Script: InsertNode-FieldDeclaration@TypeDeclaration … DeleteNode-VariableDeclarationStatement[1]**

**Group Size: 9**

**Script Length: 10**

**Relevant(Similar) Changes: 7(77.8%)**

1) insert-node@FieldDeclaration2TypeDeclaration|insert-tree@MethodDeclaration2TypeDeclaration|insert-node@Modifier2FieldDeclaration|move-tree@SimpleType2FieldDeclaration|insert-tree@VariableDeclarationFragment2FieldDeclaration|insert-node@ExpressionStatement2Block|insert-node@Assignment2ExpressionStatement|insert-node@SimpleName2Assignment|insert-node@ASSIGNMENT_OPERATOR2Assignment|move-tree@ClassInstanceCreation2Assignment|delete-node@SimpleName|delete-node@VariableDeclarationFragment|delete-node@VariableDeclarationStatement|

# EVALUATION

## c. Examples

**pdfbox/src/main/java/org/apache/pdfbox/pdmodel/encryption/PublicKeySecurityHandler.java**

```
        int sha1InputOffset = 20;
-       for(int i=0; i<recipientFieldsBytes.length; i++)
-       {
-               System.arraycopy(
-                       recipientFieldsBytes[i], 0,
-                       sha1Input, sha1InputOffset, recipientFieldsBytes[i].length);
-               sha1InputOffset += recipientFieldsBytes[i].length;
-       }
+       for (byte[] recipientFieldsByte : recipientFieldsBytes)
+       {
+               System.arraycopy(
+                       recipientFieldsByte, 0, sha1Input,
+                       sha1InputOffset, recipientFieldsByte.length);
+               sha1InputOffset += recipientFieldsByte.length;
+       }
        MessageDigest md = MessageDigests.getSHA1();
        byte[] mdResult = md.digest(sha1Input);
```

**core/src/main/java/org/apache/carbondata/core/util/ByteUtil.java**

```
byte[] flattenedData = new byte[totalSize];
        int pos = 0;
-       for (int i = 0; i < input.length; i++) {
-               System.arraycopy(input[i], 0, flattenedData, pos, input[i].length);
-               pos += input[i].length;
+       for (byte[] bytes : input) {
+               System.arraycopy(bytes, 0, flattenedData, pos, bytes.length);
+               pos += bytes.length;
        }
        return flattenedData;
```

**Script: InsertNode-EnhancedForStatement@Block … DeleteNode-ForStatement[1]**

**Group Size: 3**

**Script Length: 20**

**Relevant(Similar) Changes: 3(100%)**

1) insert-node@EnhancedForStatement2Block|insert-tree@SingleVariableDeclaration2EnhancedForStatement|insert-node@SimpleName2EnhancedForStatement|insert-node@Block2EnhancedForStatement|insert-node@ExpressionStatement2Block|insert-tree@ExpressionStatement2Block|insert-node@MethodInvocation2ExpressionStatement|move-tree@METHOD_INVOCATION_RECEIVER2MethodInvocation|insert-node@SimpleName2MethodInvocation|insert-tree@METHOD_INVOCATION_ARGUMENTS2MethodInvocation|delete-tree@VariableDeclarationExpression|delete-tree@InfixExpression|delete-tree@PostfixExpression|delete-node@SimpleName|delete-tree@METHOD_INVOCATION_ARGUMENTS|delete-node@MethodInvocation|delete-node@ExpressionStatement|delete-tree@ExpressionStatement|delete-node@Block|delete-node@ForStatement|

# CONCLUSION

# CONCLUSION

## Summary

1. Edit Script information could be extracted in various ways to find syntactic patterns of changes.
2. The recommended range of script length in the suggested method is around 1 to 20.
3. The suggested method shows the possibility of conveying semantic similarity.
4. Apart from Java, the popular research subject language, Python and C can also be patternized using the edit script information.

# CONCLUSION

## Summary

1. Edit Script information could be extracted in various ways to find syntactic patterns of changes.
2. **The recommended range of script length in the suggested method is around 1 to 20.**
3. The suggested method shows the possibility of conveying semantic similarity.
4. Apart from Java, the popular research subject language, Python and C can also be patternized using the edit script information.

# CONCLUSION

## Summary

1. Edit Script information could be extracted in various ways to find syntactic patterns of changes.
2. The recommended range of script length in the suggested method is around 1 to 20.
3. **The suggested method shows the possibility of conveying semantic similarity.**
4. Apart from Java, the popular research subject language, Python and C can also be patternized using the edit script information.

# CONCLUSION

## Summary

1. Edit Script information could be extracted in various ways to find syntactic patterns of changes.
2. The recommended range of script length in the suggested method is around 1 to 20.
3. The suggested method shows the possibility of conveying semantic similarity.
4. **Apart from Java, the popular research subject language, Python and C can also be patternized using the edit script information.**

# CONCLUSION

## Discussion & Future Work

1. By collecting specific change data(ex. BIC - BFC sets) instead of all commits, the patterns grouped by the suggested method are expected to be used for the related researches (patch-pattern survey, patch automation, patch transplantation, code suggestion, etc.).

2. The suggested method ignores similar changes with different script-order (hunk-order)

# CONCLUSION

## Discussion & Future Work

1. By collecting specific change data(ex. BIC - BFC sets) instead of all commits, the patterns grouped by the suggested method are expected to be used for the related researches (patch-pattern survey, patch automation, patch transplantation, code suggestion, etc.).

2. The suggested method ignores similar changes with different script-order (hunk-order)

# CONCLUSION

## Discussion & Future Work

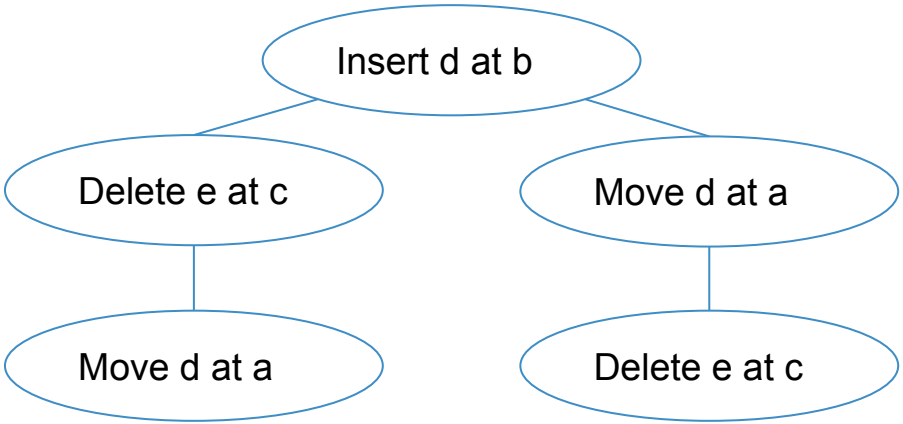
*Pattern1* = “**Insert d at b**” + “**Delete e from c**” + “**Move d at a**”

*Pattern2* = “**Insert d at b**” + “**Move d at a**” + “**Delete e from c**”

$Pattern1 \cap Pattern2 \neq \varnothing$

# CONCLUSION

## Discussion & Future Work



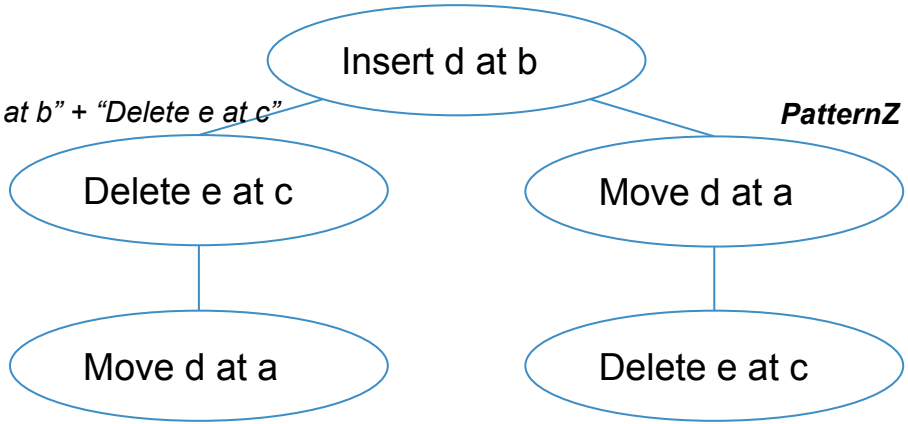

*Pattern1* = **“Insert d at b” + “Delete e from c” + “Move d at a”**

*Pattern2* = **“Insert d at b” + “Move d at a” + “Delete e from c”**

# CONCLUSION

## Discussion & Future Work

***PatternX*** = *“Insert d at b”*




***PatternY*** = *“Insert d at b” + “Delete e at c”*

***PatternZ*** = *“Insert d at b” + “Move d at a”*

***Pattern1*** = “Insert d at b” + “Delete e from c” + “Move d at a”

***Pattern2*** = “Insert d at b” + “Move d at a” + “Delete e from c”

# Q & A

## Contacts

이 창 공
EMAIL - leechanggong@gmail.com
남 재 창
EMAIL - jcnam@handong.edu