An Exploratory Study on Interface Similarities in Code Clones

Md Rakib Hossain Misu, Abduş Satter, Kazi Sakib
Institute of Information Technology
University of Dhaka
Dhaka, Bangladesh
bsse0516@iit.du.ac.bd, bit0401@iit.du.ac.bd, sakib@iit.du.ac.bd
Presentation Outline

• Introduction & Background
• Motivation
• Research Questions
• Overview of Study Design
• Experimental Dataset
• Study Results
• Discussion and Findings
• Artifacts
Introduction

- **Code Clones**
  - Similar pieces of code, within or between software systems known as code clones.
Types of Clones

- Intra-Project Clones
- Inter-Project Clones
- Type-1 (T1): Exact Clones
- Type-2 (T2): Renamed Clones
- Type-3 (T3): Gapped or Updated Clones
- Type-4 (T4): Semantic Clones
Method Interface

- Method Interfaces
  - Interface refers to the return type, method names and parameter types of a method sometimes that repeats exactly or similarly across the code repositories.

```java
int getFactorial ( int fact ){
    //check base condition
    if ( fact== 0){
        return 1;
    }
    //call recursive functino
    else{
        return fact * getFactorial (fact -1);
    }
}
```

```java
int factorial ( int n ){
    int i , fact =1;
    for ( i =1; i <= n; i ++)
        fact = fact * i ;
    return fact;
}
```
Motivation

Two major motivations of our study.

- If two methods contain the similar interface, it is very likely that they perform analogous functions either entirely or at least partially.

- If those methods contain same interface and perform similar functionality, it indicates that these methods should be semantic or syntactic code clone to each other.
Research Questions

- **RQ1**: What does percentage of interface similarities occur in intra-project and inter-project method clones with various similarity combinations?

- **RQ2**: Are the intensities of interface similarity different in various types of clones and which clone-type(s) have higher possibilities to be detected by using interface similarity?

- **RQ3**: How does interface similarity relates to code clone detection? More specifically, how many code clones occur due to interface similarity?
Overview of Study Design
Overview of Study Design

Subject Systems

Small

Medium

Large

Experimental Dataset Selection

Each Subject System
Overview of Study Design

Subject Systems
- Small
- Medium
- Large

Experimental Dataset Selection

Code Clone Detection
- Intra-Project Clone Detection
- Inter-Project Clone Detection

Each Method Clone List

Method Clone List
- method (m)

Each Method Clone

Interface Extraction

Return type, Parameter types and Keywords Extraction from Source code
- Removing Stop Words
- Transforming Words into Its Root Forms

Finding Synonyms For Each Root Word
Overview of Study Design

Subject Systems

Small

Medium

Large

Experimental Dataset Selection

Each Subject System

Code Clone Detection

Intra-Project Clone Detection

Type-1, Type-2, Type-3 Clone Detection

Inter-Project Clone Detection

Intra-Project Method Clone List

Various Types Method Clone Lists

Each Method Clone

Method Clone List

Each Method Clone

Return type, Parameter types and Keywords Extraction from Source code

Removing Stop Words

Finding Synonyms For Each Root Word

Transforming Words into Its Root Forms

Satisfying Interface Similarity conditions for each Method Clone

Interface Similarity Measurement
Experimental Dataset

- **Small Subject System (SSS)**
  - 35 open source Apache Java projects are selected as SSS

- **Medium Subject System (MSS)**
  - SF100 is a statistically sound test data generation benchmark containing 100 open source Java projects [2][3].

- **Large Subject System (LSS)**
  - IJaDataset-2.03, a large Java source code repository, covers above 24k projects crawled from GitHub, SourceForge etc. [4].
## Interface Similarity Conditions

<table>
<thead>
<tr>
<th>Identifier</th>
<th>Similarity Conditions</th>
</tr>
</thead>
<tbody>
<tr>
<td>S1</td>
<td>Return types are similar</td>
</tr>
<tr>
<td>S2</td>
<td>Number and types of parameters are similar</td>
</tr>
<tr>
<td>S3</td>
<td>At least one parameter is similar</td>
</tr>
<tr>
<td>S4</td>
<td>Return types and parameter types are similar</td>
</tr>
<tr>
<td>S5</td>
<td>Return types and at least one parameter type are similar</td>
</tr>
<tr>
<td>S6</td>
<td>At least one keyword extracted from method name is similar</td>
</tr>
<tr>
<td>S7</td>
<td>Keywords extracted from method name are similar</td>
</tr>
<tr>
<td>S8</td>
<td>At least one synonym of extracted keyword is similar</td>
</tr>
<tr>
<td>S9</td>
<td>At least one synonym from all keywords are similar</td>
</tr>
<tr>
<td>S10</td>
<td>Return types and all keywords and parameters are similar</td>
</tr>
<tr>
<td>S11</td>
<td>Return and parameter types, at least one keyword are similar</td>
</tr>
</tbody>
</table>
Study Results-RQ1

- **RQ1**: What does percentage of interface similarities occur in intra-project and inter-project method clones with various similarity combinations?

**Answer:**

- Approximately above **85%** intra-project and inter-project method clones contain similar return type and parameter types.
- **59.17%** inter-project clone contains similar keywords from method names, return type and parameter type.
Study Results-RQ2

- **RQ2**: Are the intensities of interface similarity different in various types of clones and which clone-type(s) have higher possibilities to be detected by using interface similarity?

**Answer:**

- **100%** Type-1 clone contains similar keywords from method names, return type and parameter types.
- On average **83.47%** Type-2 and **81.90%** Type-3 clones contain similar keywords from method names, return type and parameter types.
- The intensity of interface similarity is higher in Type-1 compared to Type-2 and Type-3 clones.
Study Results-RQ3

- **RQ3**: How does interface similarity relates to code clone detection? More specifically, how many code clones occur due to interface similarity?

**Answer:**
- In this case, only the intra-project clones are considered because these are the method clones that are implemented by the developers of each project.
- It is found that out of 1,85,360 intra-project method clones only 25,241 clones do not contain similar interfaces that refer only 13.62% clones.
- **86.38%** clones occur due to interface similarity. It shows interface similarity may have significant relationship to classical method clone detection.
Discussion

Reasons for why some clones do not satisfy interface similarity conditions.

- Inappropriate naming convention
- Improper term in the method name
- Type mismatch problem
- Usages of generic type
Findings and Outcomes

Findings help to design

Interface Driven Code Clone Detection
## Related Work

### Clone Detection
- According to Roy et al. clone detection techniques can be categorized into various types such String-based [5], Token-based [6], Tree-based techniques [3].
- NiCad[5], Deckard[3], SourcererCC[8]

### Code Search
- Keyword Based Code Search (KBCS) [8], Semantic Based Code Search (SBCS) [10], and Test Driven Code Search (TDCS) [21].

### Interface Redundancy
- Interface Redundancy (IR) represents the repetition of whole method interface (e.g., return type, method name, and parameters types) across the software corpus [13].
- 80% project of the targeted repositories contain redundant interfaces.
- It is observed that IR has diverged from traditional code cloning since in their study only 0.002% IR is related to method clones.
Artifacts

- Source Code of Interface Similarity Detection
  - https://github.com/MisuBeImp/CloneInterfaceSimilarityDetector
- Paper Artifacts Detected Clones, Source of Subject Systems
  - https://github.com/MisuBeImp/APSEC-2017-Paper-Artifacts
Thank You
References


Backup Slides
Type-1 (T1) Clones

Identical code fragments, except for differences in white-space, layout, and comments

```
1 int factorial ( int n ){
2   if ( n == 0)
3       return 1;
4   else
5       return n * factorial (n -1);
6 }
```
Type-2 (T2) Clones

Identical code fragments, except for differences in identifier names and literal values, in addition to Type-1 clone differences.

```c
1 int factorial ( int n ){
2    if ( n == 0)
3        return 1;
4    else
5        return n * factorial (n -1);
6 }

1 int findFactorial ( int num){
2    if (num == 0)
3        return 1;
4    else
5        return num * findFactorial (num-1);
6 }
```
Type-3 (T3) Clones

Syntactically similar code fragments that differ at the statement level. The fragments have statements added, modified and/or removed with respect to each other, in addition to Type-1 and Type-2 clone differences.
Type-4 (T4) Clones

Syntactically dissimilar code fragments that implement the same functionality. They are also known as semantic or functional clones.

```c
int factorial ( int n ){
    if ( n == 0)
        return 1;
    else
        return n * factorial (n -1);
}
```
Why Do Code Clones Exist?

1) Cloning as a Way to Reuse
2) Cloning for Maintenance Benefits
3) Limitation of Programming Languages/Frameworks
4) Software Development Practices
5) Cloning by Chance (Accidental Cloning)
Issues Due to Code Cloning

1) Increase the probability of bug propagation
2) Cloning a code fragment can be error prone and may introduce new bugs in the system
3) Difficult for locating and fixing possible bugs.
4) Increase the size of a software
5) Break design abstractions or indicate lack of inheritance
Applications of Clone Detection

1) License Violation and Copyright Infringement
2) Code Search
3) Reverse Engineering Product Line Architecture
4) Plagiarism Detection
5) Library or API detection
6) Software Provenance Analysis
7) Multi-version Program Analysis
8) Program Understanding