1st WETSoDA , December 4, 2017 - Nanjing, China

An Exploratory Study on Interface Similarities in Code Clones

Md Rakib Hossain Misu, Abdus 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

Presented By

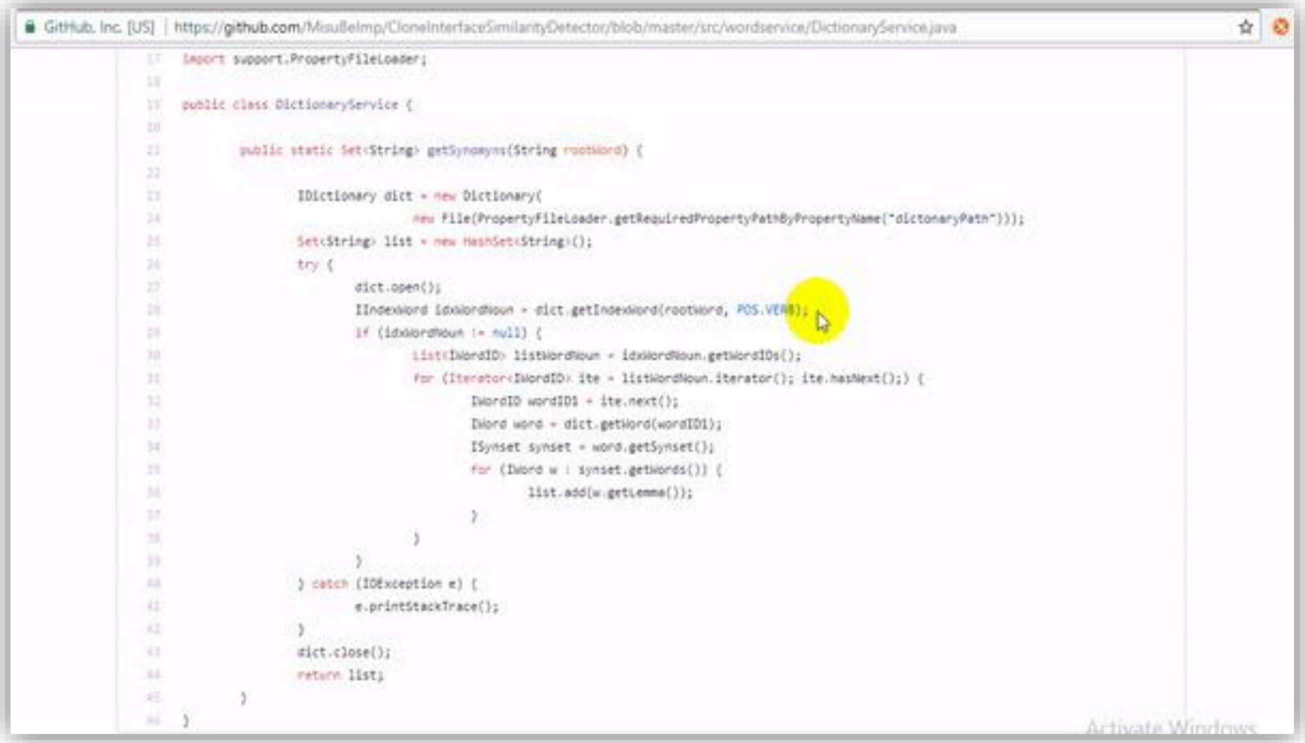
Md Rakib Hossain Misu
Institute of Information Technology
University of Dhaka
Dhaka, Bangladesh

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.



GitHub, Inc. [US] | https://github.com/MisuBeImp/CloneInterfaceSimilarityDetector/blob/master/src/wordservice/DictionaryService.java

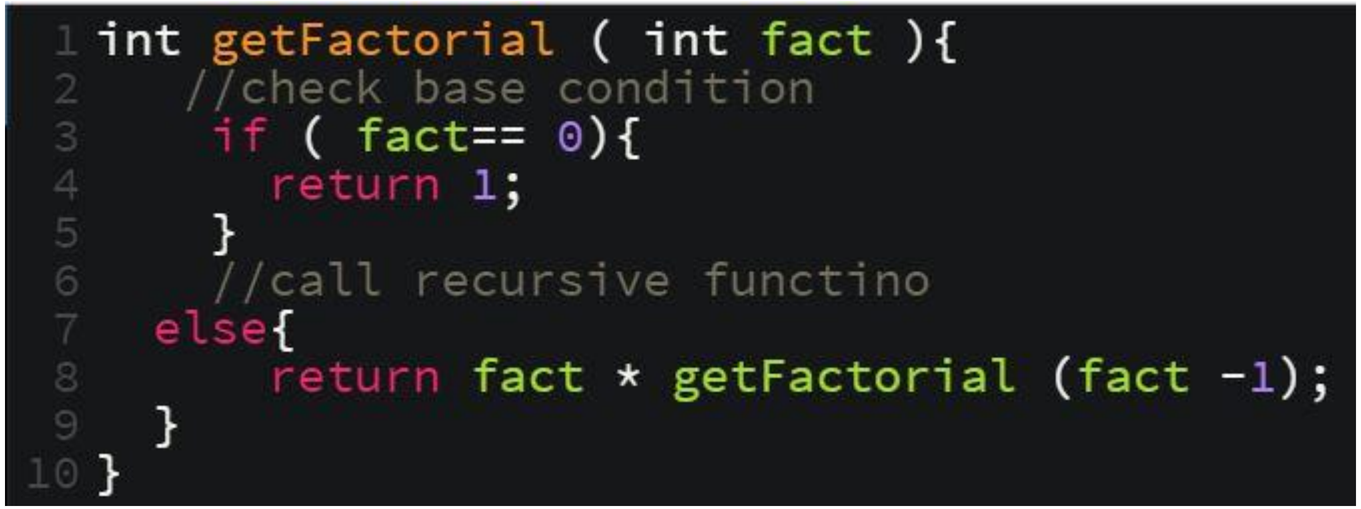
```
import support.PropertyFileLoader;

public class DictionaryService {

        public static Set<String> getSynomyns(String rootWord) {

                IDictionary dict = new Dictionary(
                                new File(PropertyFileLoader.getRequiredPropertyPathByPropertyName("dictonaryPath")));
                Set<String> list = new HashSet<String>();
                try {
                        dict.open();
                        IIndexWord idxWordNoun = dict.getIndexWord(rootWord, POS.VERB);
                        if (idxWordNoun != null) {
                                List<IWordID> listWordNoun = idxWordNoun.getWordIDs();
                                for (Iterator<IWordID> ite = listWordNoun.iterator(); ite.hasNext();) {
                                        IWordID wordID1 = ite.next();
                                        IWord word = dict.getWord(wordID1);
                                        ISynset synset = word.getSynset();
                                        for (IWord w : synset.getWords()) {
                                                list.add(w.getLemma());
                                        }
                                }
                        }
                } catch (IOException e) {
                        e.printStackTrace();
                }
                dict.close();
                return list;
        }
}
```

Types of Clones

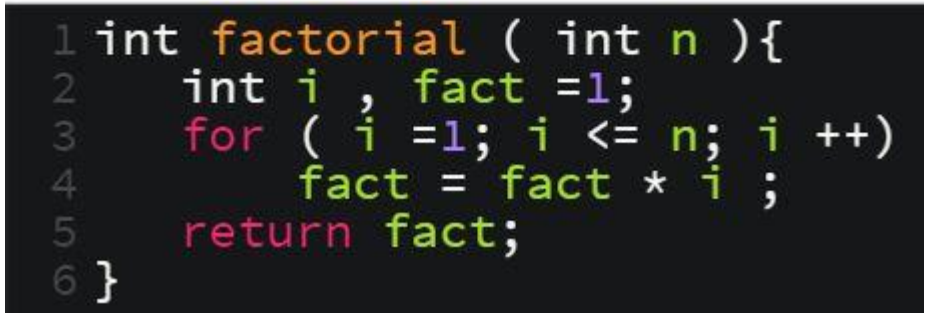
- Intra-Project Clones
- Inter-Project Clones
- Type-1 (T1): Exact Clones
- Type-2 (T2): Renamed Clones
- Type-3 (T3): Gapped or Updated Clones
- Type-4 (T4): Semantic Clones

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



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



```
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

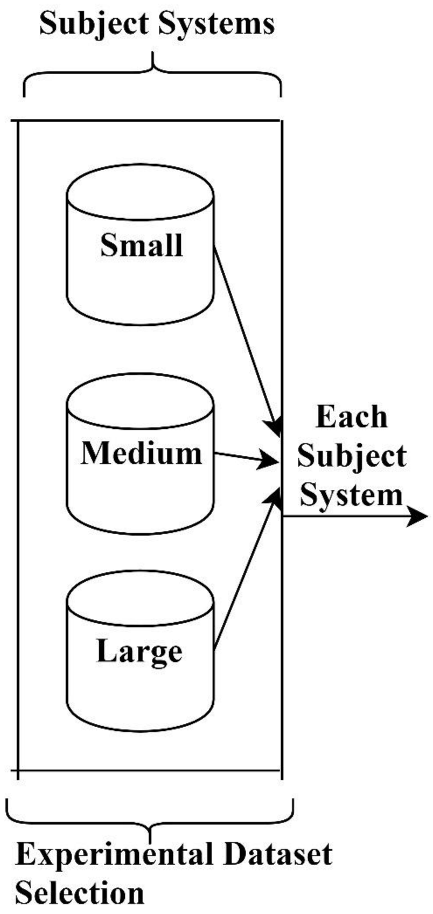
- **RQ1:** What does percentage of interface similarities occur in intra-project and inter-project method clones with various similarity combinations?

- **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?

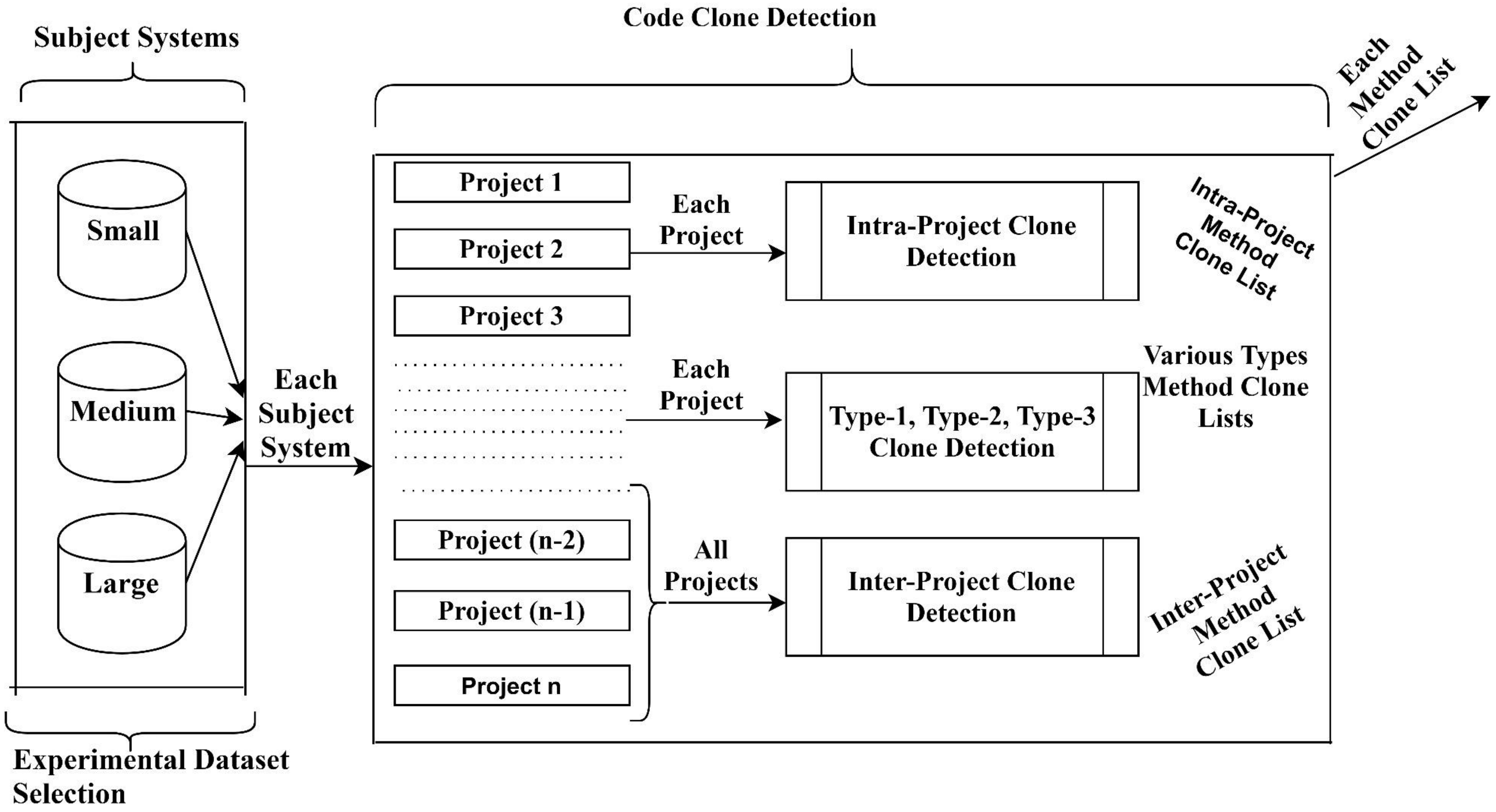
- **RQ3:** How does interface similarity relates to code clone detection? More specifically, how many code clones occur due to interface similarity?

Overview of Study Design

Overview of Study Design

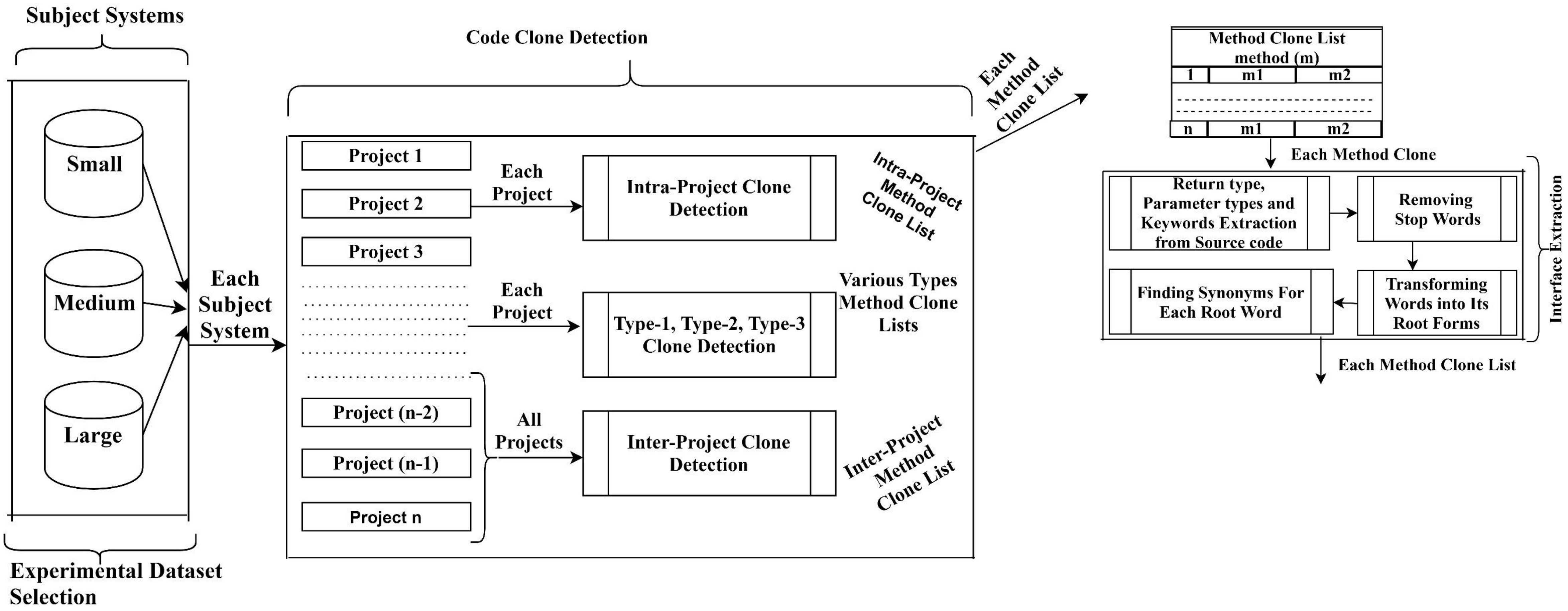


Overview of Study Design



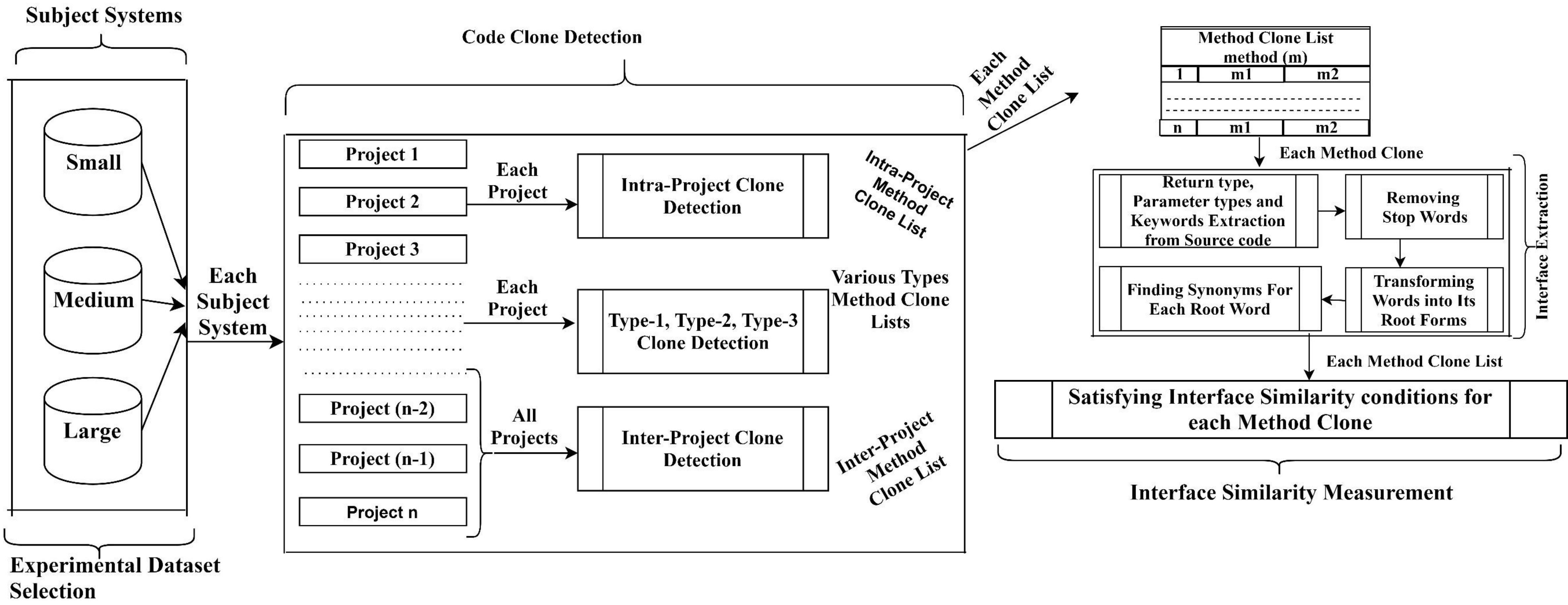
Subject Systems
Small
Medium
Large
Experimental Dataset Selection
Each Subject System
Code Clone Detection
Project 1
Project 2
Project 3
Project (n-2)
Project (n-1)
Project n
Each Project
Intra-Project Clone Detection
Intra-Project Method Clone List
Each Project
Type-1, Type-2, Type-3 Clone Detection
Various Types Method Clone Lists
All Projects
Inter-Project Clone Detection
Inter-Project Method Clone List
Each Method Clone List

Overview of Study Design


Subject Systems
Small
Medium
Large
Experimental Dataset Selection
Each Subject System
Code Clone Detection
Project 1
Project 2
Project 3
Project (n-2)
Project (n-1)
Project n
Each Project
Intra-Project Clone Detection
Each Project
Type-1, Type-2, Type-3 Clone Detection
All Projects
Inter-Project Clone Detection
Intra-Project Method Clone List
Various Types Method Clone Lists
Inter-Project Method Clone List
Each Method Clone List
Each Method Clone
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
Interface Extraction
Each Method Clone List

Method Clone List method (m)		
1	m1	m2
n	m1	m2

Overview of Study Design


Subject Systems
Small
Medium
Large
Experimental Dataset Selection
Each Subject System
Code Clone Detection
Project 1
Project 2
Project 3
Project (n-2)
Project (n-1)
Project n
Each Project
Intra-Project Clone Detection
Intra-Project Method Clone List
Each Project
Type-1, Type-2, Type-3 Clone Detection
Various Types Method Clone Lists
All Projects
Inter-Project Clone Detection
Inter-Project Method Clone List
Each Method Clone List
Method Clone List
Each Method Clone
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
Interface Extraction
Each Method Clone List
Satisfying Interface Similarity conditions for each Method Clone
Interface Similarity Measurement

	method (m)	
1	m1	m2
n	m1	m2

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

Identifier	Similarity Conditions
S1	Return types are similar
S2	Number and types of parameters are similar
S3	At least one parameter is similar
S4	Return types and parameter types are similar
S5	Return types and at least one parameter type are similar
S6	At least one keyword extracted from method name is similar
S7	Keywords extracted from method name are similar
S8	At least one synonym of extracted keyword is similar
S9	At least one synonym from all keywords are similar
S10	Return types and all keywords and parameters are similar
S11	Return and parameter types, at least one keyword are similar

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 contains 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

Finings 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 Driven Code Search (IDCS) [11]. It allows users to search code in code libraries, by using interface information.

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

[1] *C. K. Roy and J. R. Cordy, "A survey on software clone detection research," Queens School of Computing TR, vol. 541, no. 115, pp. 64– 68, 2007.*

[2] *G. Fraser and A. Arcuri, "Sound empirical evidence in software testing," in Software Engineering (ICSE), 2012 34th International Conference on. IEEE, 2012, pp. 178–188*

[3] *G. Fraser and A. Arcuri "A large-scale evaluation of automated unit test generation using evosuite," ACM Transactions on Software Engineering and Methodology (TOSEM), vol. 24, no. 2, p. 8, 2014.*

[4] *J. Svajlenko and C. K. Roy, "Bigcloneeval: A clone detection tool evaluation framework with bigclonebench," in Software Maintenance and Evolution (ICSME), 2016 IEEE International Conference on. IEEE, 2016, pp. 596–600.*

[5] *C. K. Roy, J. R. Cordy, and R. Koschke, "Comparison and evaluation of code clone detection techniques and tools: A qualitative approach," Science of Computer Programming, vol. 74, no. 7, pp. 470–495, 2009*

[6] *J. R. Cordy and C. K. Roy, "The nicad clone detector," in Program Comprehension (ICPC), 2011 IEEE 19th International Conference on. IEEE, 2011, pp. 219–220*

[7] *L. Jiang, G. Misherghi, Z. Su, and S. Glondu, "Deckard: Scalable and accurate tree-based detection of code clones," in Proceedings of the 29th international conference on Software Engineering. IEEE Computer Society, 2007, pp. 96–105*

[8] *H. Sajnani, V. Saini, J. Svajlenko, C. K. Roy, and C. V. Lopes, "Sourcerercc: scaling code clone detection to big-code," in Proceedings of the 38th International Conference on Software Engineering. ACM, 2016, pp. 1157–1168.*

References

[9] *W. B. Frakes and B. A. Nejmeh, "Software reuse through information retrieval," in ACM SIGIR Forum, vol. 21, no. 1-2. ACM, 1986, pp. 30–36*

[10] *S. P. Reiss, "Semantics-based code search," in Proceedings of the 31st International Conference on Software Engineering. IEEE Computer Society, 2009, pp. 243–253*

[11] *O. A. Lazzarini Lemos, S. K. Bajracharya, and J. Ossher, "Codegenie:: a tool for test-driven source code search," in Companion to the 22nd ACM SIGPLAN conference on Object-oriented programming systems and applications companion. ACM, 2007, pp. 917–918*

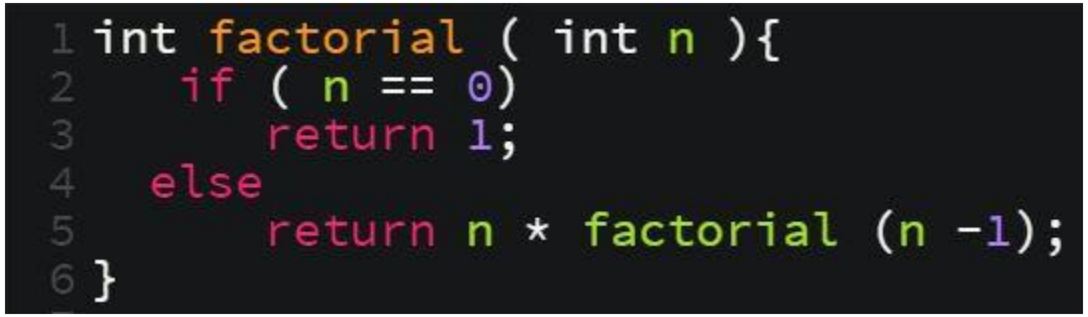
[12] *A. M. Zaremski and J. M. Wing, "Signature matching: a tool for using software libraries," ACM Transactions on Software Engineering and Methodology (TOSEM), vol. 4, no. 2, pp. 146–170, 1995.*

[13] *A. C. de Paula, E. Guerra, C. V. Lopes, H. Sajnani, and O. A. L. Lemos, "An exploratory study of interface redundancy in code repositories," in Source Code Analysis and Manipulation (SCAM), 2016 IEEE 16th International Working Conference on. IEEE, 2016, pp. 107–116*

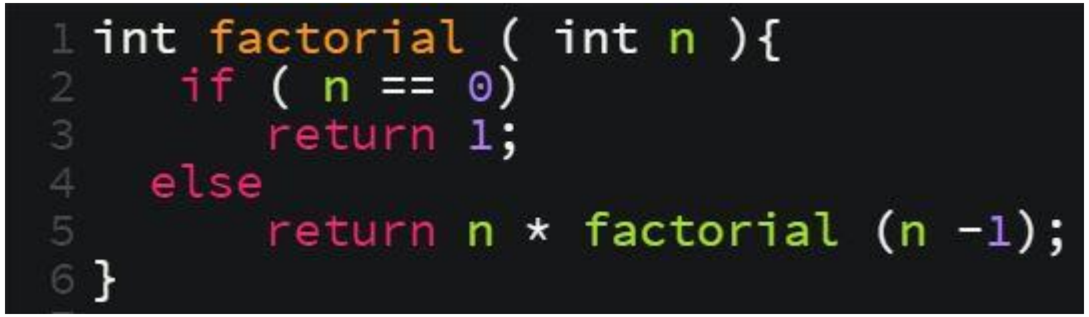
Backup Slides

Type-1 (T1) Clones

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



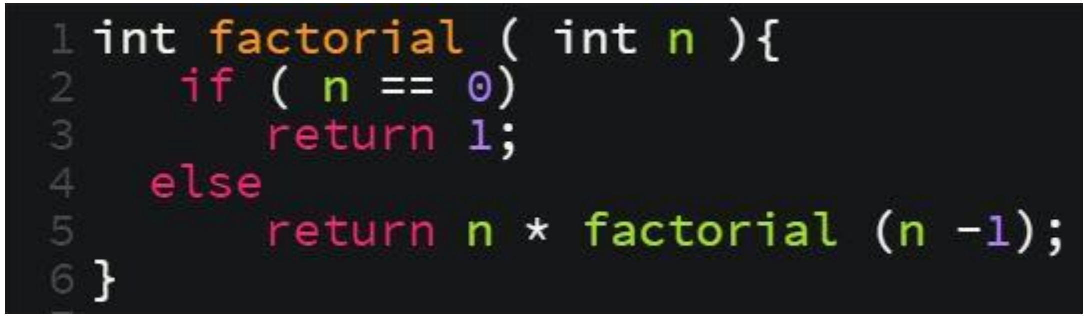
```
int factorial ( int n ){
   if ( n == 0)
      return 1;
  else
      return n * factorial (n -1);
}
```



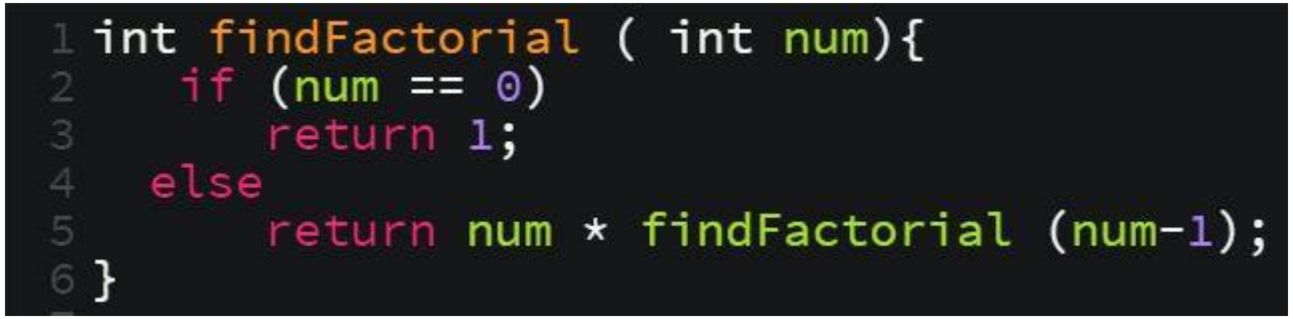
```
int factorial ( int n ){
   if ( n == 0)
      return 1;
  else
      return n * factorial (n -1);
}
```

Type-2 (T2) Clones

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



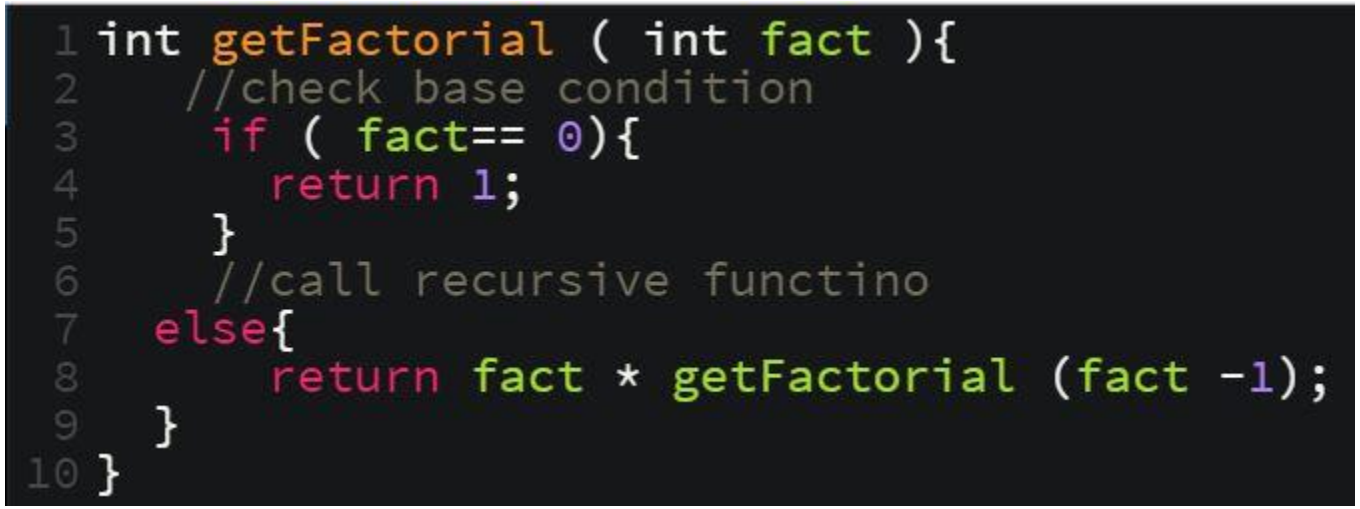
```
int factorial ( int n ){
   if ( n == 0)
      return 1;
  else
      return n * factorial (n -1);
}
```



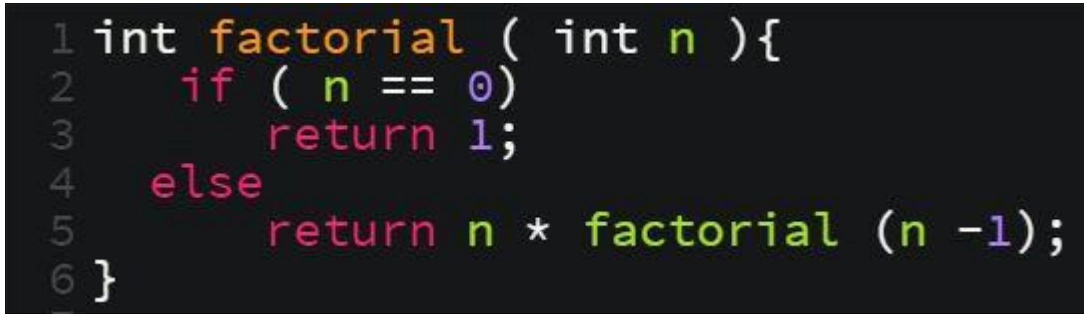
```
int findFactorial ( int num){
   if (num == 0)
      return 1;
  else
      return num * findFactorial (num-1);
}
```

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



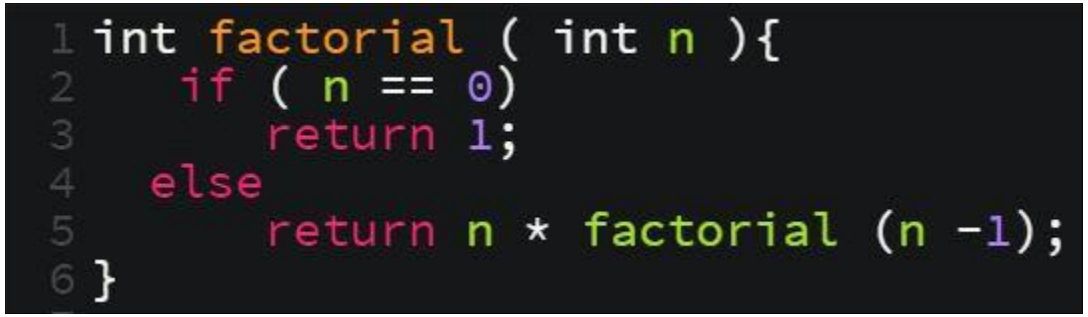
```
int getFactorial ( int fact ){
   //check base condition
    if ( fact== 0){
      return 1;
    }
    //call recursive functino
  else{
      return fact * getFactorial (fact -1);
  }
}
```



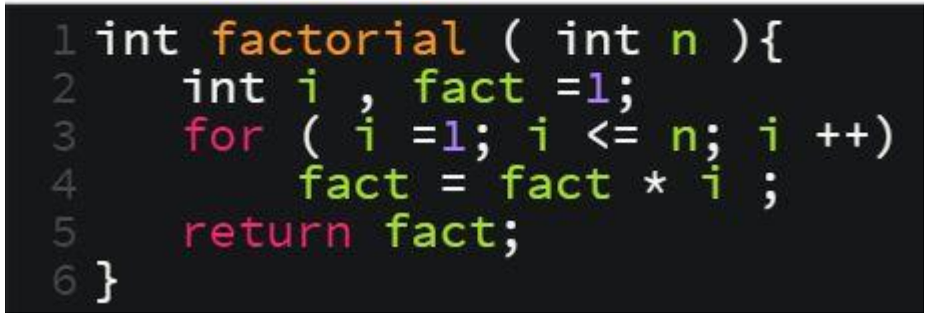
```
int factorial ( int n ){
   if ( n == 0)
      return 1;
  else
      return n * factorial (n -1);
}
```

Type-4 (T4) Clones

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



```
int factorial ( int n ){
   if ( n == 0)
      return 1;
  else
      return n * factorial (n -1);
}
```



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

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