

## Code Clone Analysis and Application

Katsuro Inoue  
Osaka University



Software Engineering Laboratory, Department of Computer Science, Graduate School of Information Science and Technology, Osaka University

## Talk Structure

- Clone Detection
- CCFinder and Associate Tools
- Applications
- Summary of Code Clone Analysis and Application



Software Engineering Laboratory, Department of Computer Science, Graduate School of Information Science and Technology, Osaka University

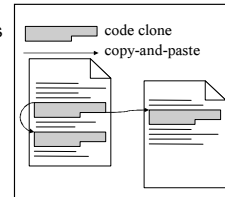
## Clone Detection



Software Engineering Laboratory, Department of Computer Science, Graduate School of Information Science and Technology, Osaka University

## What is Code Clone?

- A code fragment which has identical or similar code fragments in source code
- Introduced in source code because of various reasons
  - code reuse by 'copy-and-paste'
  - stereotyped function
    - ex. file open, DB connect, ...
  - intentional iteration
    - performance enhancement
- It makes software maintenance more difficult
  - If we modify a code clone with many similar code fragments, it is necessary to consider whether or not we have to modify each of them
    - It is likely to overlook



Software Engineering Laboratory, Department of Computer Science, Graduate School of Information Science and Technology, Osaka University

## Simple Example

```
AFG::AFG(JaObject* obj) {  
    objname = "afg";  
    object = obj;  
}  
AFG::~AFG() {  
    for(unsigned int i=0; i< children.size(); i++)  
        if(children[i] != NULL)  
            delete children[i];  
    ...  
    for(unsigned int i=0;  
        i< nodes.size(); i++)  
        if(nodes[i] != NULL)  
            delete nodes[i];  
}
```



Software Engineering Laboratory, Department of Computer Science, Graduate School of Information Science and Technology, Osaka University

## Definition of Code Clone

- No single or generic definition of code clone
  - So far, several methods of code clone detection have been proposed, and each of them has its own definition about code clone
- Various detection methods
  1. Line-based comparison
  2. AST (Abstract Syntax Tree) based comparison
  3. PDG (Program Dependency Graph) based comparison
  4. Metrics comparison
  5. Token-based comparison



Software Engineering Laboratory, Department of Computer Science, Graduate School of Information Science and Technology, Osaka University

Detection Method

### 1. Line-Based Comparison

- Detect code clone by comparing source code on line unit[1]
  - Before comparison, tabs and white-spaces are eliminated
- This is a method of an early days
- Detection accuracy is low
  - Cannot detect code clones written in different coding styles
    - ex. '{' position of if-statement or while-statement
  - Cannot detect code clones using different variable names
    - we want to identify the same logic code as code clones even if variable names are different

[1] B. S. Baker, *A Program for Identifying Duplicated Code*, Proc. Computing Science and Statistics 24<sup>th</sup> Symposium on the Interface, pp.49-57, Mar. 1992.



Software Engineering Laboratory, Department of Computer Science, Graduate School of Information Science and Technology, Osaka University

Detection Method

### 2. AST Based Comparison

- Parse source code, and construct AST (Abstract Syntax Tree)
  - Similar subtrees are identified as code clones[2]
    - The differences of code style and variable name are eliminated
- Fairly practical method
  - Commercial tool

CloneDR:

<http://www.semanticdesigns.com/Products/Clone/>

[2] D. Baxter, A. Yahin, L. Moura, M.S. Anna, and L. Bier, *Clone Detection Using Abstract Syntax Trees*, Proc. International Conference on Software Maintenance 98, pp.368-377, 16-19, Nov. 1998.



Software Engineering Laboratory, Department of Computer Science, Graduate School of Information Science and Technology, Osaka University

Detection Method

### 3. PDG Based Comparison

- Build PDG (Program Dependence Graph) using the result of semantic analysis
  - Similar sub-graphs are identified as code clones [3]
- The detection accuracy is very high
- Can detect code clones which are not detected in other methods
  - semantic clone, reordered clone
- Require complex computation
  - It is very difficult to apply to large software

[3] R. Komondoor and S. Horwitz, *Using slicing to identify duplication in source code*, Proc. the 8th International Symposium on Static Analysis, pp.40-56, July, 16-18, 2001.



Software Engineering Laboratory, Department of Computer Science, Graduate School of Information Science and Technology, Osaka University

Detection Method

### 4. Metrics Comparison

- Calculate metrics for each function unit
  - Units with the similar metrics values are identified as code clones [4]
- Partly similar units are not detected
- Suitable to large scale analysis

[4] J. Mayland, C. Leblanc, and E.M. Merlo, *Experiment on the automatic detection of function clones in a software system using metrics*, Proc. International Conference on Software Maintenance 96, pp.244-253, Nov. 1996.



Software Engineering Laboratory, Department of Computer Science, Graduate School of Information Science and Technology, Osaka University

Detection Method

### 5. Token Based Comparison

- Compare token sequences of source code, and identify the similar subsequence as code clones[5]
  - Before comparison, tokens of identifier (type name, variable name, method name, ...) are replaced by the same special token (parameterization)
- The Scalability is very high
  - M Loc / 5-20 min.

[5] T. Kamiya, S. Kusumoto, and K. Inoue, *CCFinder: A multi-linguistic token-based code clone detection system for large scale source code*, IEEE Transactions on Software Engineering, vol. 28, no. 7, pp. 654-670, Jul. 2002.



Software Engineering Laboratory, Department of Computer Science, Graduate School of Information Science and Technology, Osaka University

## CCFinder and Associate Tools

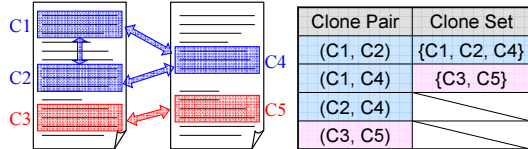
---



Software Engineering Laboratory, Department of Computer Science, Graduate School of Information Science and Technology, Osaka University

### Clone Pair and Clone Set

- Clone Pair
  - a pair of identical or similar code fragments
- Clone Set
  - a set of identical or similar fragments



Software Engineering Laboratory, Department of Computer Science, Graduate School of Information Science and Technology, Osaka University

### Our Code Clone Research

- Develop tools
  - Detection tool: CCFinder
  - Visualization tool: Gemini
  - Refactoring support tool: Ariès
  - Change support tool: Libra
- Deliver our tools to domestic or overseas organizations/individuals
  - More than 100 companies uses our tools!
- Promote academic-industrial collaboration
  - Organize code clone seminars
  - Manage mailing-lists

Software Engineering Laboratory, Department of Computer Science, Graduate School of Information Science and Technology, Osaka University

Detection tool:

### Development of CCFinder

- Developed by industry requirement
  - Maintenance of a huge system
    - More than 10M LOC, more than 20 years old
    - Maintenance of code clones by hand had been performed, but ...
- Token-base clone detection tool CCFinder
  - Normalization of name space
  - Parameterization of user-defined names
  - Removal of table initialization
  - Identification of module delimiter
  - Suffix-tree algorithm
- CCFinder can analyze the system of millions line scale in 5-30 min.

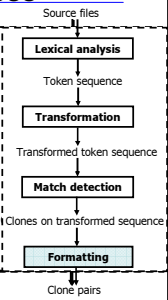
Software Engineering Laboratory, Department of Computer Science, Graduate School of Information Science and Technology, Osaka University

Detection tool:

### CCFinder Detection Process

```

1. static void foo() throws RE:SyntaxException {
2.   String a[] = new String[] { "123.400", "abc", "orange 100" };
3.   org.apache.regex.RE pat = new org.apache.regex.RE("(0-9,)+");
4.   int sum = 0;
5.   for (int i = 0; i < a.length; ++i)
6.     if (pat.match(a[i]))
7.       sum += Sample.parseNumber(pat.getParent(0));
8.   System.out.println("sum = " + sum);
9. }
10. static void gook(String[] a) throws RE:SyntaxException {
11.   RE exp = new RE("(0-9,)+");
12.   int sum = 0;
13.   for (int i = 0; i < a.length; ++i)
14.     if (exp.match(a[i]))
15.       sum += parseNumber(exp.getParent(0));
16.   System.out.println("sum = " + sum);
17. }
    
```

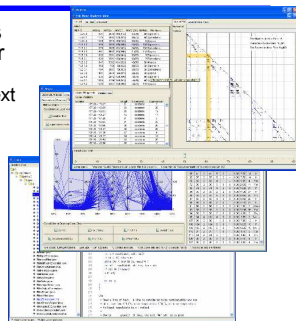


Software Engineering Laboratory, Department of Computer Science, Graduate School of Information Science and Technology, Osaka University

Visualization Tool:

### Gemini Outline

- Visualize code clones detected by CCFinder
  - CCFinder outputs the detection result to a text file
- Provide interactive analyses of code clones
  - Scatter Plot
  - Clone metrics
  - File metrics
- Filter out unimportant code clones

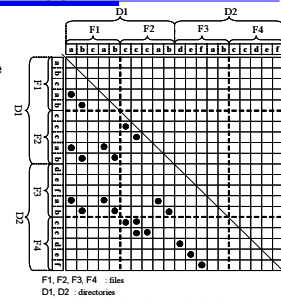


Software Engineering Laboratory, Department of Computer Science, Graduate School of Information Science and Technology, Osaka University

Visualization tool:

### Gemini Scatter Plot


- Visually show where code clones are
- Both the vertical and horizontal axes represent the token sequence of source code
  - The original point is the upper left corner
- Dot means corresponding two tokens on the two axes are the same
  - Symmetric to main diagonal (show only lower left)



Software Engineering Laboratory, Department of Computer Science, Graduate School of Information Science and Technology, Osaka University

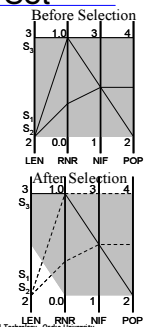

Visualization tool:  
**Gemini Clone and File Metrics**

- Metrics are used to quantitatively characterize entities
- Clone metrics
  - $LEN(S)$ : the average length of code fragments (the number of tokens) in clone set  $S$
  - $POP(S)$ : the number of code fragments in  $S$
  - $NIF(S)$ : the number of source files including any fragments of  $S$
  - $RNR(S)$ : the ratio of non-repeated code sequence in  $S$
- File metrics
  - $ROC(F)$ : the ratio of duplication of file  $F$ 
    - if completely duplicated, the value is 1.0
    - if not duplicated at all, the value is 0,0
  - $NOF(F)$ : the number of code fragments of any clone set in file  $F$
  - $NOF(F)$ : the number of files sharing any code clones with file  $F$




Visualization tool:  
**Gemini Selection of Clone Set**

- We introduced selection mechanism, **Metric Graph**
  - Each metric has parallel coordinate axes
  - A polygonal line is drawn per clone set
- The user can specify the upper and lower limits of each metric
  - The hatching part is the range bounded by the upper and lower limit
  - A clone set is *selected* state if its all metric values are within the range
  - The user can easily browse source code of selected code clones

**Refactoring Support System: Aries (1)**

- Structural code clones are regarded as the target of refactoring
  - Detect clone pairs by CCFinder
  - Transform the detected clone pairs into clone sets
  - Extract structural parts as structural code clones from the detected clone sets
- What is structural code clone ?
  - example: Java language
    - Declaration: class declaration, interface declaration
    - Method: method body, constructor, static initializer
    - statement: do, for, if, switch, synchronized, try, while

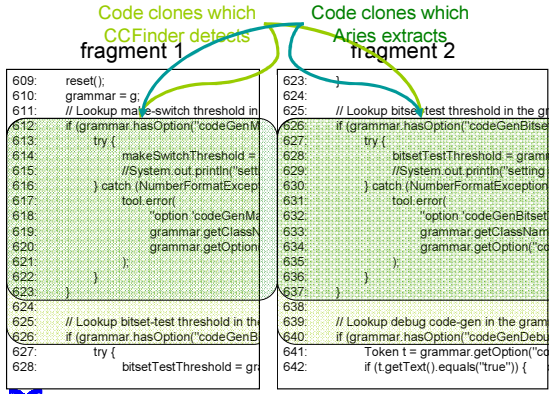



Code clones which CCFinder detects fragment 1

Code clones which Aries extracts fragment 2

```


609: reset();
610: grammar = g;
611: // Lookup make-switch threshold in
612: if (grammar.hasOption("codeGenM
613: try {
614:     makeSwitchThreshold =
615:     //System.out.println("sett
616:     } catch (NumberFormatException
617:     tool.error();
618:     *option "codeGenM
619:     grammar.getClassN
620:     grammar.getOption
621:     };
622: }
623: }
624:
625: // Lookup bitset-test threshold in th
626: if (grammar.hasOption("codeGenB
627: try {
628:     bitsetTestThreshold = gr
629:
630:
631:
632:
633:
634:
635:
636:
637: }
638:
639: // Lookup debug code-gen in the gram
640: if (grammar.hasOption("codeGenDebu
641:     Token t = grammar.getOption("cd
642:     if (t.getText().equals("true")) {
643:
644:
645:
646:
647:
648:
649:
650:
651:
652:
653:
654:
655:
656:
657:
658:
659:
660:
661:
662:
663:
664:
665:
666:
667:
668:
669:
670:
671:
672:
673:
674:
675:
676:
677:
678:
679:
680:
681:
682:
683:
684:
685:
686:
687:
688:
689:
690:
691:
692:
693:
694:
695:
696:
697:
698:
699:
700:
    
```

**Refactoring Support System: Aries (2)**

- Following refactoring patterns[1][2] can be used to remove code sets including structural code clones
  - Extract Class,
  - Extract Method,
  - Extract Super Class,
  - Form Template Method,
  - Move Method,
  - Parameterize Method,
  - Pull Up Constructor,
  - Pull Up Method,
- For each clone set, Aries suggests which refactoring pattern is applicable by using metrics.

[1]: M. Fowler: Refactoring: Improving the Design of Existing Code, Addison-Wesley, 1999.  
 [2]: <http://www.refactoring.com/>, 2004.



**Refactoring**

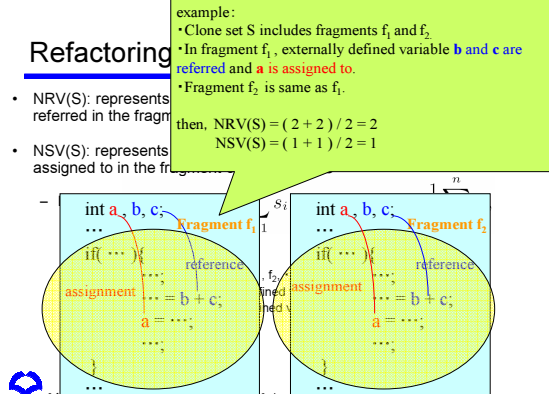

example:

- Clone set  $S$  includes fragments  $f_1$  and  $f_2$
- In fragment  $f_1$ , externally defined variable  $b$  and  $c$  are referred and  $a$  is assigned to.
- Fragment  $f_2$  is same as  $f_1$ .

NRV(S): represents referred in the fragment

NSV(S): represents assigned to in the fragment

then,  $NRV(S) = (2 + 2) / 2 = 2$   
 $NSV(S) = (1 + 1) / 2 = 1$







Academic-industrial collaboration:

## Code Clone Seminar

- We have periodically organized code clone seminars from Dec 2002
- Seminar is the place to exchange views with industrial people
- Seminar overview
  - Tool demonstration
  - Lecture of how to use code clone information
  - Case study of companies using our tools



Software Engineering Laboratory, Department of Computer Science, Graduate School of Information Science and Technology, Osaka University

## Case Studies

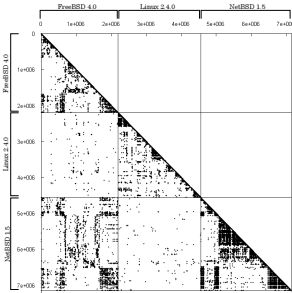
- Open source software
  - FreeBSD, NetBSD, Linux(C, 7MLOC)
  - JDK Libraries(Java 1.8MLOC)
  - Qt(C++, 240KLOC)
- Commercial software (more than 100 companies)
  - IPA/SEC, NTT Data Corp., Hitachi Ltd., Hitachi GP, Hitachi SAS, NEC soft Ltd., ASTEC Inc., SRA Inc., JAXA, Daiwa Computer, etc...
- Students excise of Osaka University
- Court evidence for software copyright suit

Software Engineering Laboratory, Department of Computer Science, Graduate School of Information Science and Technology, Osaka University

Case study 1:

## Similarity between FreeBSD, NetBSD, Linux

- Result
  - There are many code clones between FreeBSD and NetBSD
  - There are a little code clones between Linux and FreeBSD/NetBSD
- Their histories can explain the result
  - The ancestors of FreeBSD and NetBSD are the same
  - Linux was made from scratch



Software Engineering Laboratory, Department of Computer Science, Graduate School of Information Science and Technology, Osaka University

Case study 2:

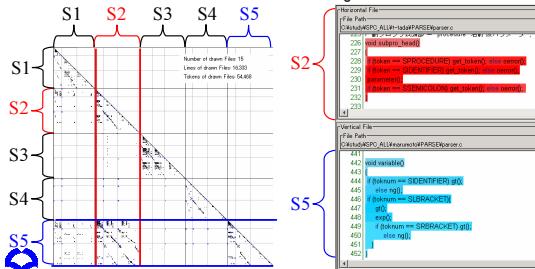
## Students Excise

- Target
  - Programs developed on a programming exercise in Osaka Univ.
    - Simple compiler for Pascal written in C language
  - This exercise consists of 3 steps
    - STEP1: develop a **syntax checker**
    - STEP2: develop a **semantics checker** by extending his/her syntax checker
    - STEP3: develop a **total compiler** by extending his/her semantics checker
- Purpose
  - Check the stepwise development
  - Check plagiarisms

Software Engineering Laboratory, Department of Computer Science, Graduate School of Information Science and Technology, Osaka University

## Result

- There were a lot of code clones between S2 and S5
- We did not use the detection result for evaluating their excises



Software Engineering Laboratory, Department of Computer Science, Graduate School of Information Science and Technology, Osaka University

Case study 3:

## IPA/SEC Advanced Project

- Target
  - A car-traffic information system using heterogeneous sensors, developed by 5 Japanese companies
  - The project manager had little knowledge of the source code since each company independently developed the components
- Purpose
  - Grasp features of black-boxed source code
- Approach
  - Analyzed twice, after the unit test (280,000LOC), and after the combined test (300,000LOC)
  - The minimum size of detected code clone is 30 tokens

Software Engineering Laboratory, Department of Computer Science, Graduate School of Information Science and Technology, Osaka University

IPA/SEC Advanced Project:  
**Duplicated Ratio**

- The below graph illustrates the distribution of duplicated ratio of the sub-system developed by a company

- We interviewed developers of the sub-system
  - They added library code to the system to add new functions right before combined test

Software Engineering Laboratory, Department of Computer Science, Graduate School of Information Science and Technology, Osaka University

IPA/SEC Advanced Project:  
**Scatter Plot Analysis**

- Scatter Plot of company X
- In part A, there are many non-interesting code clones
  - output code for debug (consecutive printf-statements)
  - check data validity
  - consecutive if-statements
- In part B, there are many code clones across directories
  - This part treats vehicle position information
  - Each directory include a single kind of vehicles, e.g., taxi, bus, or truck
  - Logical structures are mostly the same

Software Engineering Laboratory, Department of Computer Science, Graduate School of Information Science and Technology, Osaka University

## Summary of Code Clone Analysis and Application

Software Engineering Laboratory, Department of Computer Science, Graduate School of Information Science and Technology, Osaka University

## Conclusion

- We have developed Code clone analysis tools
  - Detection tool: CCFinder
  - Visualization tool: Gemini
  - Refactoring support tool: Aries
  - Debug support tool: Libra
- We have promoted academic-industrial collaboration
  - Organize code clone seminars
  - Manage mailing lists
- We have applied our tools to various software

Software Engineering Laboratory, Department of Computer Science, Graduate School of Information Science and Technology, Osaka University

## Future Direction

- CCFinderX by Kamiya at AIST Japan.
  - Token analyzer is definable
- System analysis via code clones associated with other metrics
- Architecture evolution by the view of code clones

Software Engineering Laboratory, Department of Computer Science, Graduate School of Information Science and Technology, Osaka University


## Resources

- Papers
  - T. Kamiya, S. Kusumoto, and K. Inoue, CCFinder: A multi-linguistic token-based code clone detection system for large scale source code, IEEE Transactions on Software Engineering, vol. 28, no. 7, pp. 654-670, Jul. 2002.
  - Many Others ... See our home page
- Web
  - CCFinder:  
<http://sel.ist.osaka-u.ac.jp/cdtools/index-e.html>
  - CCFinderX:  
<http://www.ccfinder.net/ccfinderx.html>
- Tools
  - See home pages

Software Engineering Laboratory, Department of Computer Science, Graduate School of Information Science and Technology, Osaka University

END

---



Software Engineering Laboratory, Department of Computer Science, Graduate School of Information Science and Technology, Osaka University

