Code Clone Analysis and Application

Katsuro Inoue

Osaka University

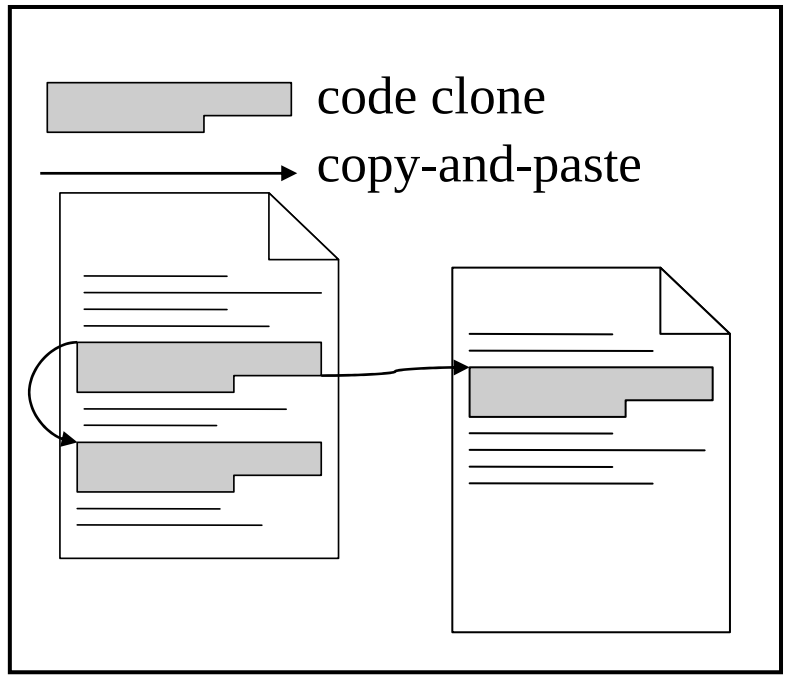
Talk Structure

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

Clone Detection

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

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];
}
```

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

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 24th Symposium on the Interface, pp.49-57, Mar. 1992.

2. AST Based Comparison

- Parse source code, and construct AST (Abstract Syntax Tree)
  - Similar sutrees 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] I.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, pp368-377, 16-19, Nov. 1998.

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.

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.

5. Token Based Comparison

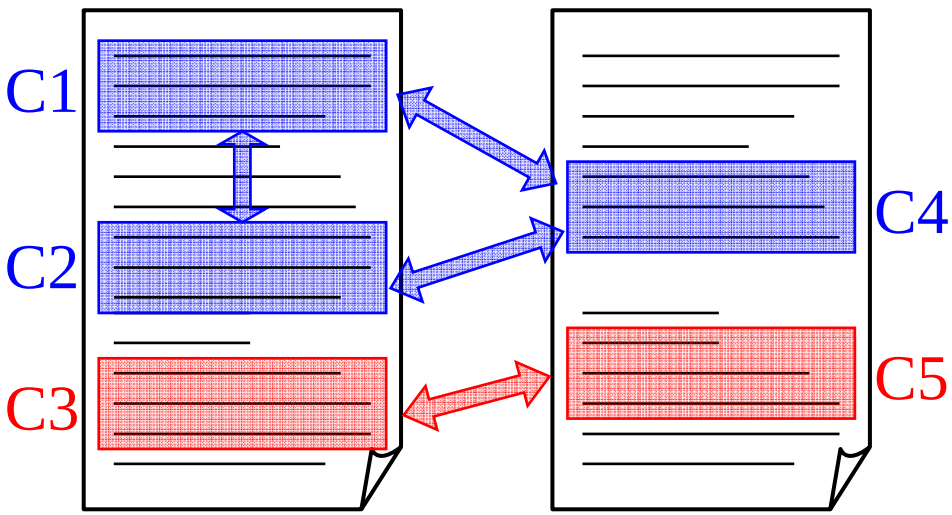
- Compare token sequences of source code, and identify the similar subsequence as code clones[5]
  - Before comparison, tokens of identifier (type name, variable name, method name, …) are replaced by the same special token (parameterization)
- The Scalability is very high
  - M Loc / 5-20 min.

[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.

CCFinder and Associate Tools

Clone Pair and Clone Set

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



Clone Pair	Clone Set
(C1, C2)	{C1, C2, C4}
(C1, C4)	{C3, C5}
(C2, C4)	
(C3, C5)	

Our Code Clone Research

- Develop tools
  - Detection tool: CCFinder
  - Visualization tool: Gemini
  - Refactoring support tool: Aries
  - 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

Detection tool:

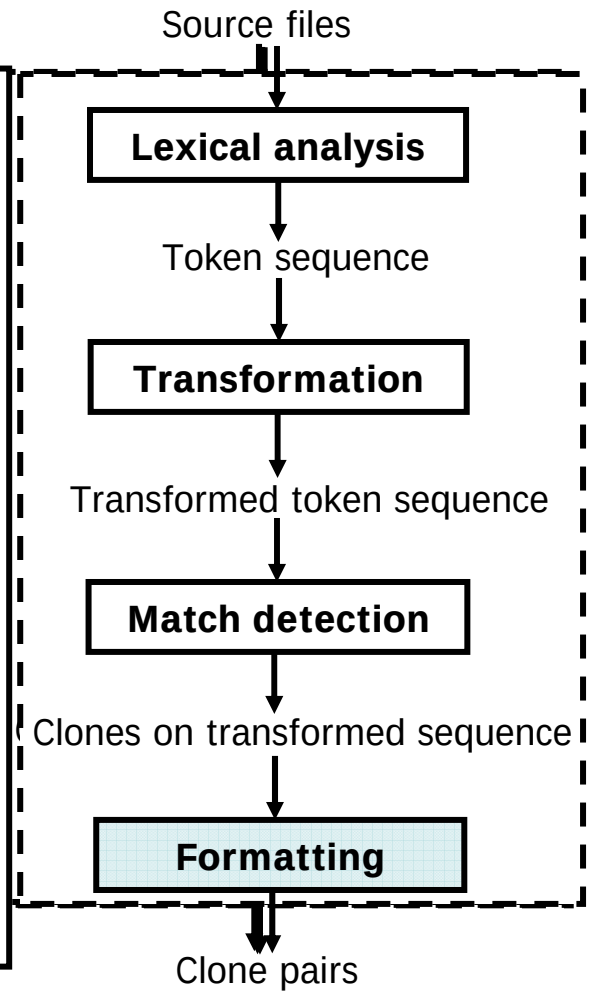
Development of CCFinder

- 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.

Detection tool:

CCFinder Detection Process

```
1. static void foo() throws RESyntaxException {
2.    String a[] = new String [] { "123,400", "abc", "orange 100" };
3.    org.apache.regexp.RE pat = new org.apache.regexp.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.getParen(0));
8.    System.out.println("sum = " + sum);
9. }
10. static void goo(String [] a) throws RESyntaxException {
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.getParen(0));
16.    System.out.println("sum = " + sum);
17. }
```



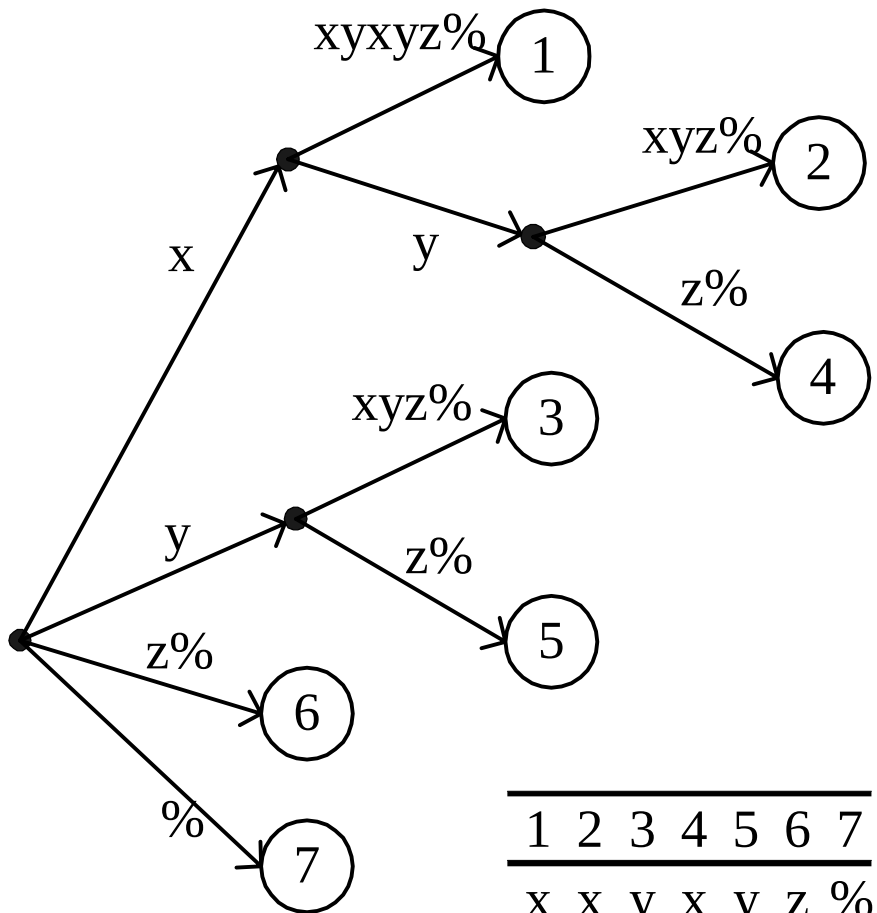
Suffix-tree

- Suffix tree is a tree that satisfies the following conditions.

1. A leaf node represents the starting position of sub-string.
2. A path from root node to a leaf node represents a sub-string.
3. First characters of labels of all the edges from one node are different from each other.

→ **A common path means a clone**

		1	2	3	4	5	6	7
		x	x	y	x	y	z	%
1	x	*						
2	x	*	*					
3	y			*				
4	x	*	*		*			
5	y			*		*		
6	z						*	
7	%							*

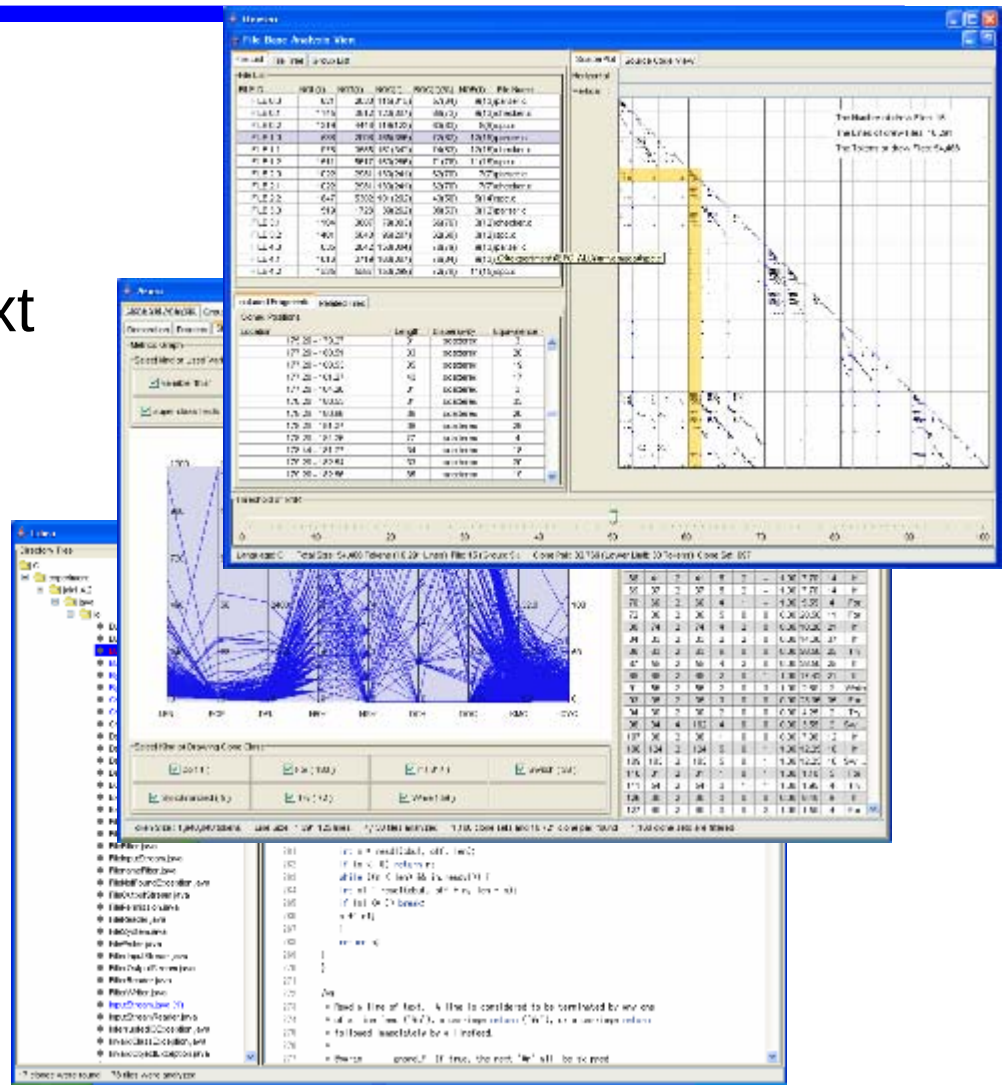


1	2	3	4	5	6	7
x	x	y	x	y	z	%

Visualization Tool:

Gemini Outline

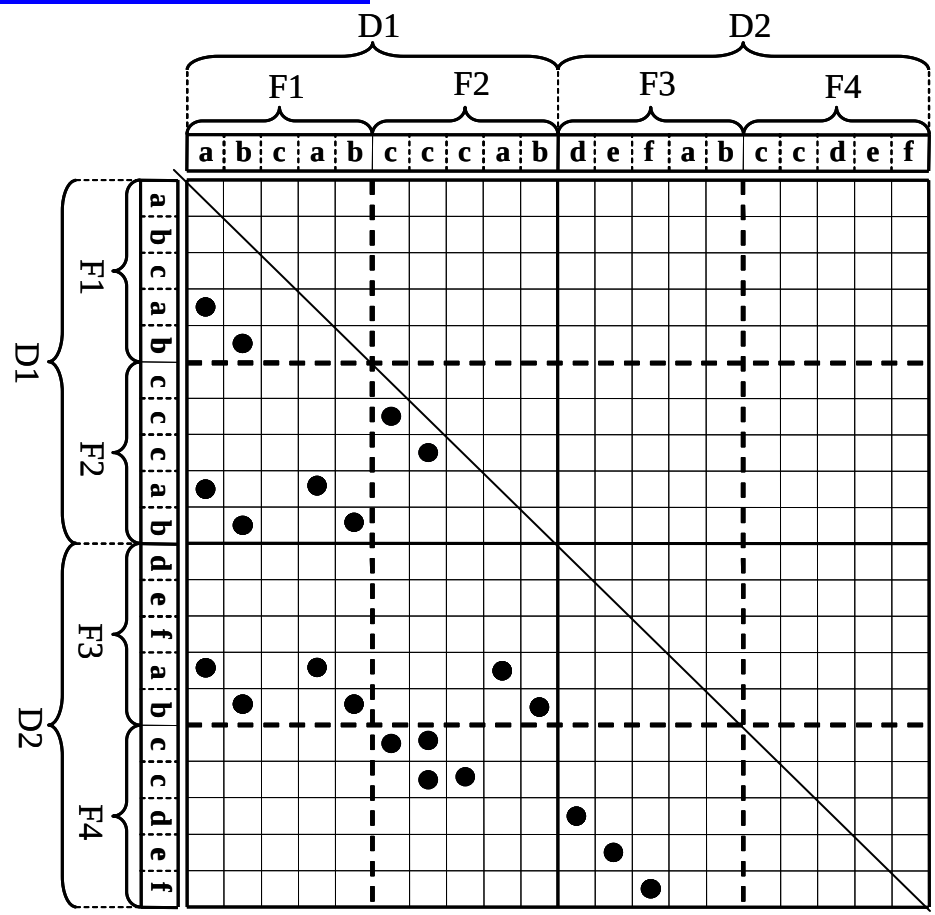
- Visualize code clones detected by CCFinder
  - CCFinder outputs the detection result to a text file
- Provide interactive analyses of code clones
  - Scatter Plot
  - Clone metrics
  - File metrics
- Filter out unimportant code clones



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)



F1, F2, F3, F4 : files

D1, D2 : directories

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
  - ***NOC(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 Metric RNR

- By a lot of experience, we fount that CCFinder detects a lot of code clones from monotonous or repetitive fragment
  - consecutive entries of switch-statements
  - consecutive variable declarations or method invocations
- Filtering metric: RNR(S)
  - Represents the ratio of non-repeated code sequence in S

Definition

$$RNR(S) = 1 - \frac{\sum_{C \in S} Tokens_{repeated}(C)}{\sum_{C \in S} Tokens_{all}(C)}$$

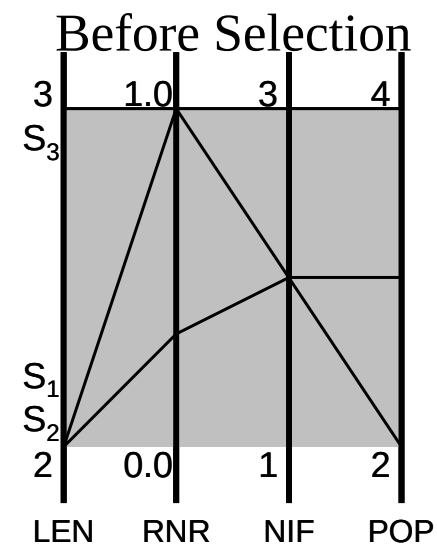
$Tokens_{all}(C)$: Number of tokens in code fragment C

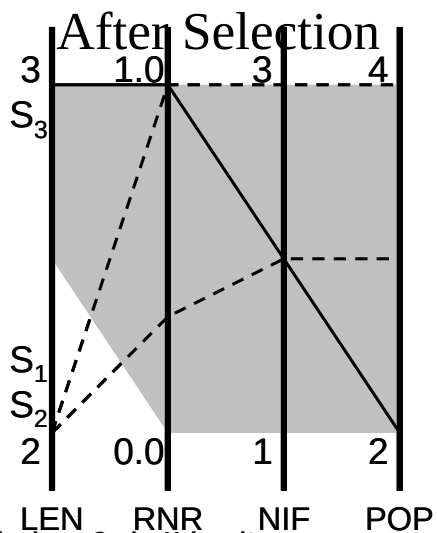
$Tokens_{repeated}(C)$: Number of repeated tokens in C

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
  1. Detect clone pairs by CCFinder
  2. Transform the detected clone pairs into clone sets
  3. 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

Code clones which Aries extracts

fragment 1

```
609:    reset();
610:    grammar = g;
611:    // Lookup ma e-switch threshold in
612:    if (grammar.hasOption("codeGenM
613:        try {
614:            makeSwitchThreshold =
615:            //System.out.println("sett
616:        } catch (NumberFormatExcep
617:            tool.error(
618:                "option 'codeGenMa
619:                grammar.getClassN
620:                grammar.getOption(
621:            );
622:        }
623:    }
624:
625:    // Lookup bitset-test threshold in the
626:    if (grammar.hasOption("codeGenB
627:        try {
628:            bitsetTestThreshold = gra
```

fragment 2

```
623:    }
624:
625:    // Lookup bitse test threshold in the gr
626:    if (grammar.hasOption("codeGenBitse
627:        try {
628:            bitsetTestThreshold = gram
629:            //System.out.println("setting
630:        } catch (NumberFormatException
631:            tool.error(
632:                "option 'codeGenBitset
633:                grammar.getClassNam
634:                grammar.getOption("co
635:            );
636:        }
637:    }
638:
639:    // Lookup debug code-gen in the gram
640:    if (grammar.hasOption("codeGenDebu
641:        Token t = grammar.getOption("co
642:        if (t.getText().equals("true")) {
```

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

- NRV(S): represents
referred in the fragm

- NSV(S): represents
assigned to in the fr

example：

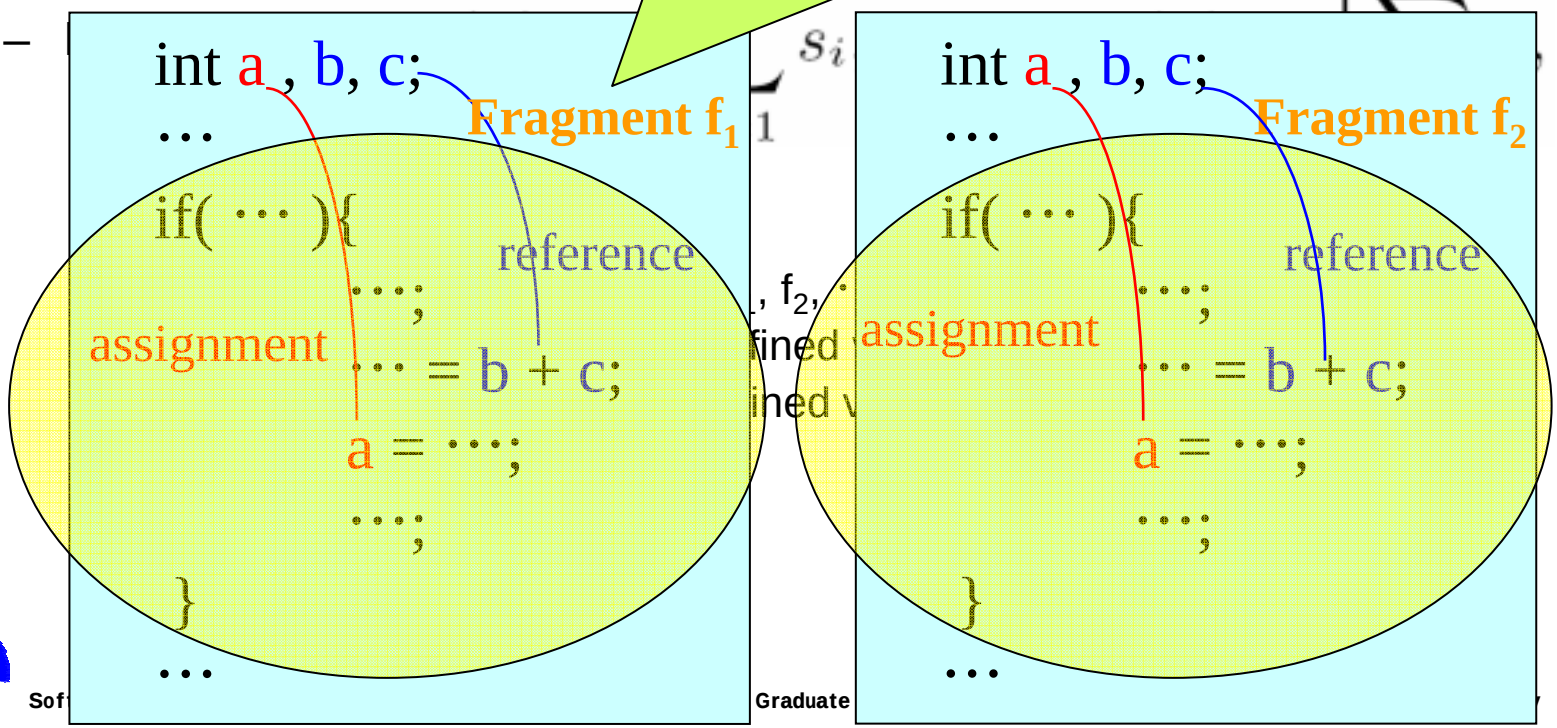
・Clone set S includes fragments f_1 and f_2.

・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.

then, NRV(S) = (2 + 2) / 2 = 2

NSV(S) = (1 + 1) / 2 = 1



Refactoring Supp

- DCH(S): represents t
  fragment of a clone s
  - Definition

$$\{D(C_1, C_p), \cdots, D(C_n, C_p)\}$$

t $f_1, f_2, \cdots, f_n$

tes lowest position in $C_1, C_2, \cdots, C_n$ on class

$C_1, C_2, \cdots, C_n$ exists, the value of DCH(S) is

nly the class hierarchy where target software exists.

class A	class B
fragment f_1	fragment f_2

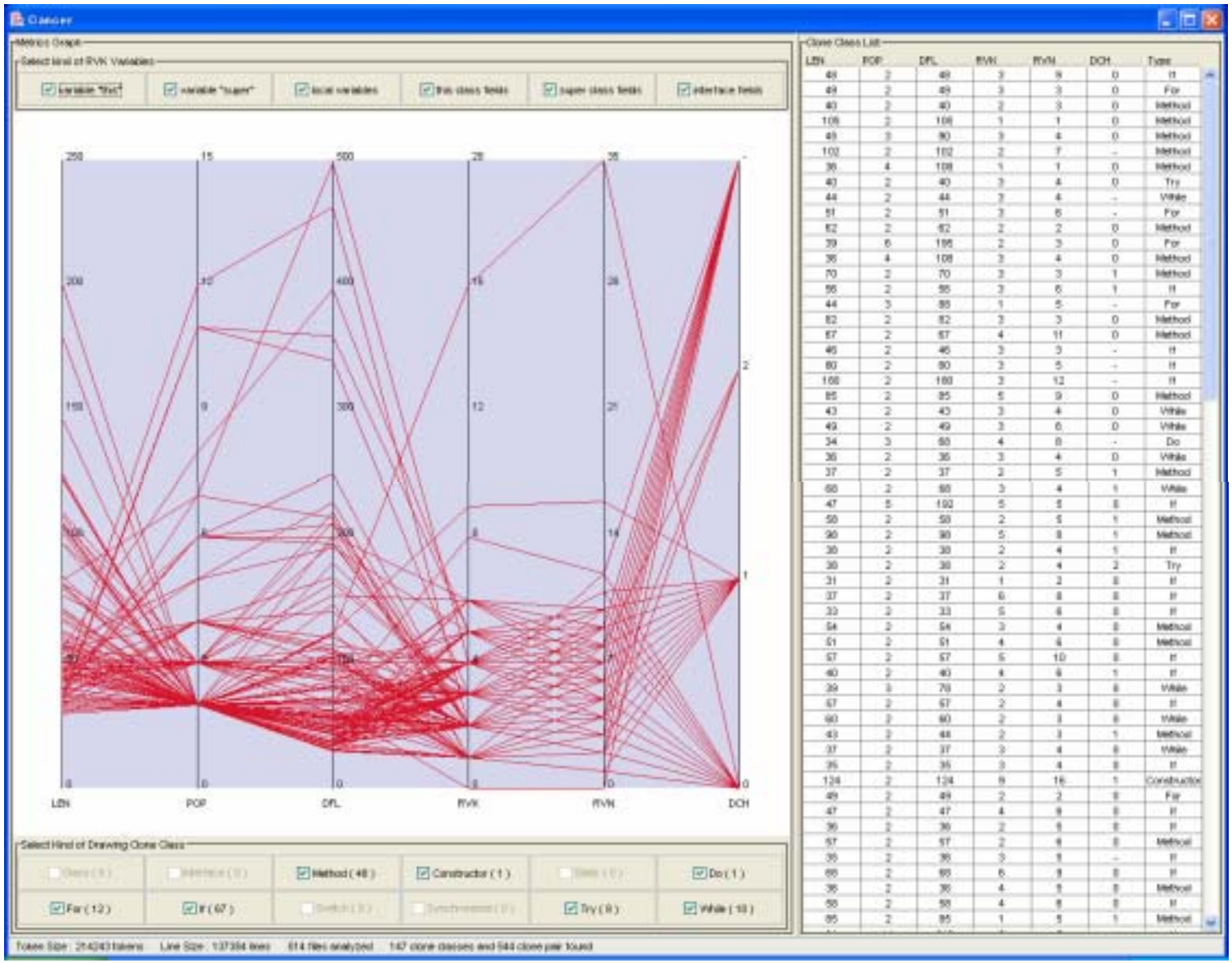
class B	class C
fragment f_1	fragment f_2

example 3

・Clone set S includes fragments f_1 and f_2.

・If all classes which include f_1 and f_2 don't have common parent class,

then, DCH(S) = ∞


Clonger
Metrics Graph
Select kind of RVK Variables
Clone Class List
LEN
POP
DFL
RVK
RVN
DCH
Type
Select Kind of Drawing Clone Class
Method (48)
Constructor (1)
Do (1)
For (12)
If (67)
Try (8)
While (18)

Clone Class Viewer

Metrics Information

Kind :If	LEN :36	POP :11	DFL :360	RVK :5	RVN :9	DCH :-

Fragment List

Path	Location	Length
C:\EXPERI~1\ant1.5.4\src\main\org\apache\tools\ant\taskdefs\optional\extension\Extension.java	584.9 - 590.9	36
C:\EXPERI~1\ant1.5.4\src\main\org\apache\tools\ant\taskdefs\optional\extension\Extension.java	592.9 - 598.9	36
C:\EXPERI~1\ant1.5.4\src\main\org\apache\tools\ant\taskdefs\optional\extension\Extension.java	600.9 - 606.9	36
C:\EXPERI~1\ant1.5.4\src\main\org\apache\tools\ant\taskdefs\optional\extension\Extension.java	608.9 - 614.9	36
C:\EXPERI~1\ant1.5.4\src\main\org\apache\tools\ant\taskdefs\optional\extension\Extension.java	616.9 - 622.9	36
C:\EXPERI~1\ant1.5.4\src\main\org\apache\tools\ant\taskdefs\optional\extension\Extension.java	624.9 - 630.9	36
C:\EXPERI~1\ant1.5.4\src\main\org\apache\tools\ant\taskdefs\optional\extension\Specification.java	474.9 - 480.9	36
C:\EXPERI~1\ant1.5.4\src\main\org\apache\tools\ant\taskdefs\optional\extension\Specification.java	482.9 - 488.9	36
C:\EXPERI~1\ant1.5.4\src\main\org\apache\tools\ant\taskdefs\optional\extension\Specification.java	490.9 - 496.9	36
C:\EXPERI~1\ant1.5.4\src\main\org\apache\tools\ant\taskdefs\optional\extension\Specification.java	498.9 - 504.9	36
C:\EXPERI~1\ant1.5.4\src\main\org\apache\tools\ant\taskdefs\optional\extension\Specification.java	506.9 - 512.9	36

Source Code View

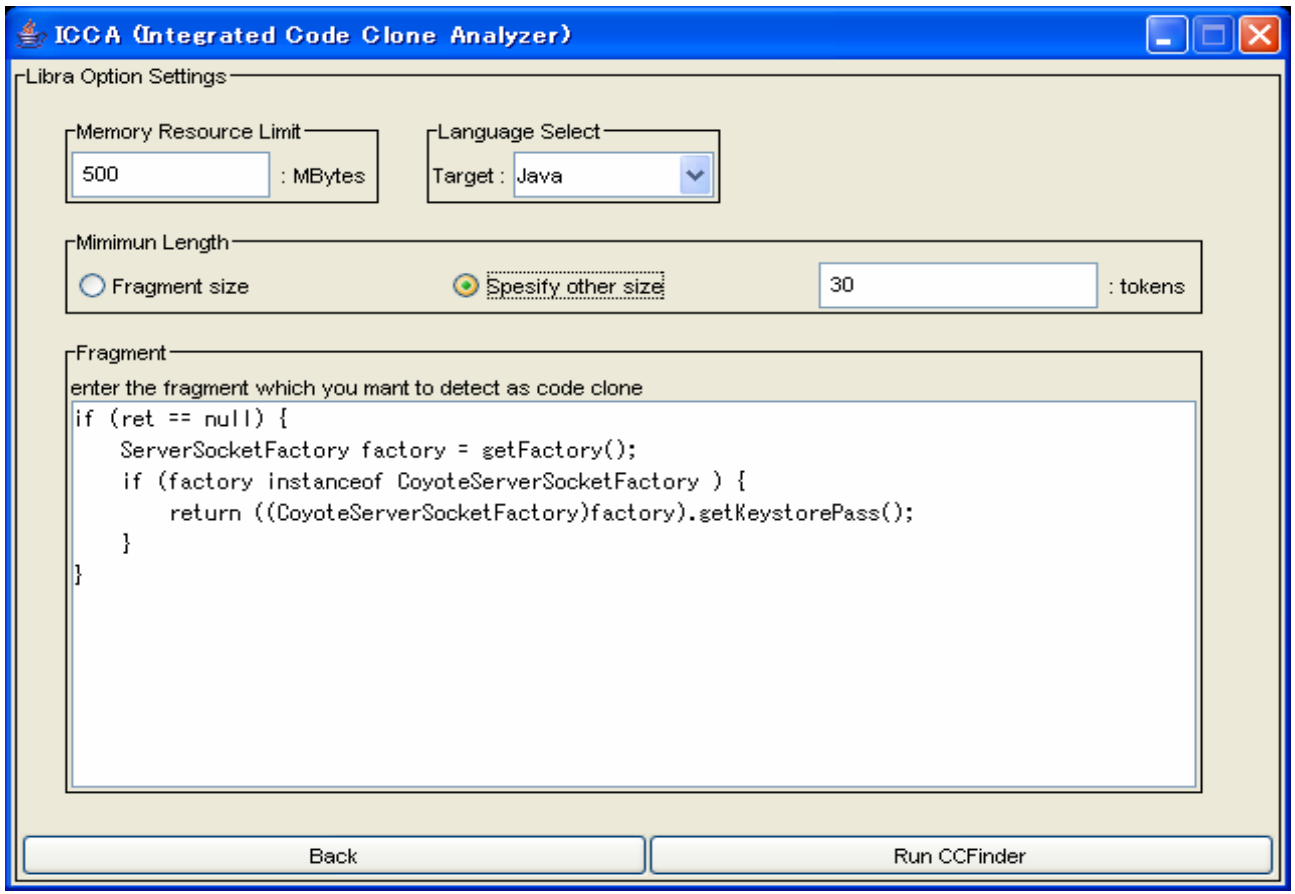
```
575     {
576         final String lineSeparator = System.getProperty( "line.separato
577         final String brace = ": ";
578
579         final StringBuffer sb = new StringBuffer( EXTENSION_NAME.toStri
580         sb.append( brace );
581         sb.append( m_extensionName );
582         sb.append( lineSeparator );
583
584         if( null != m_specificationVersion )
585         {
586             sb.append( SPECIFICATION_VERSION );
587             sb.append( brace );
588             sb.append( m_specificationVersion );
589             sb.append( lineSeparator );
590         }
591
```

Variable List

Name	Declared Location	Referred Time
SPECIFICATION_VERSION	this_class_field	1
brace	local	1
lineSeparator	local	1
m_specificationVersion	this_class_field	2
sb	local	4

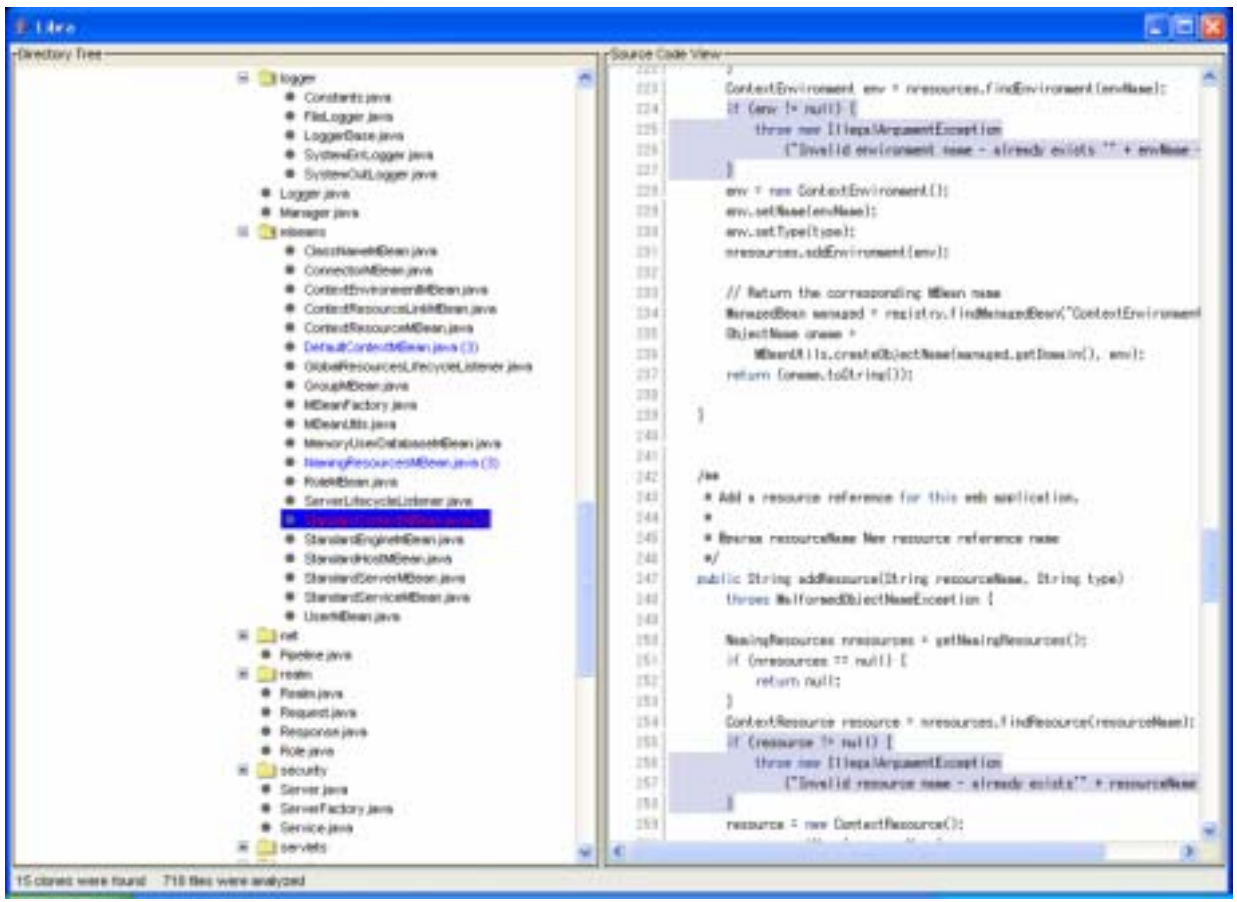
Change Support System: Libra

- Input a code fragment



Change Support System: Libra (2)

- Find clones between the input and target



Applications

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

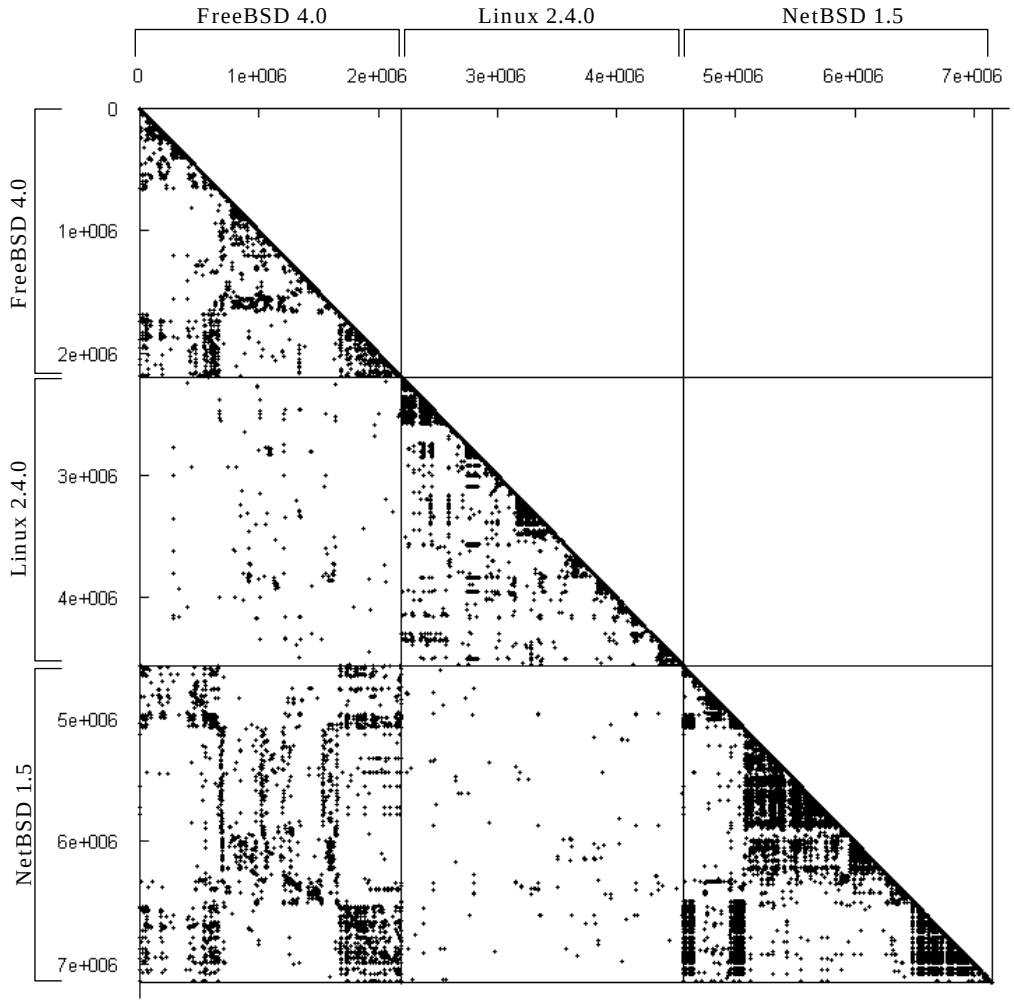
Case Studies

- 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

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



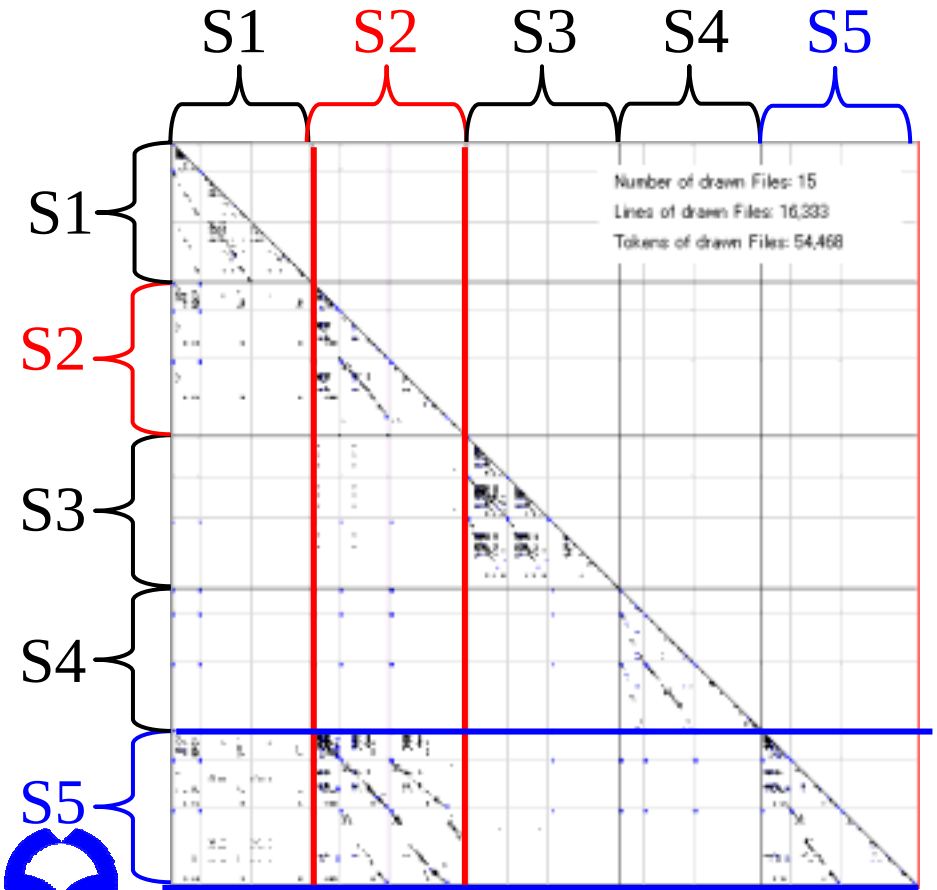
Case study 2:

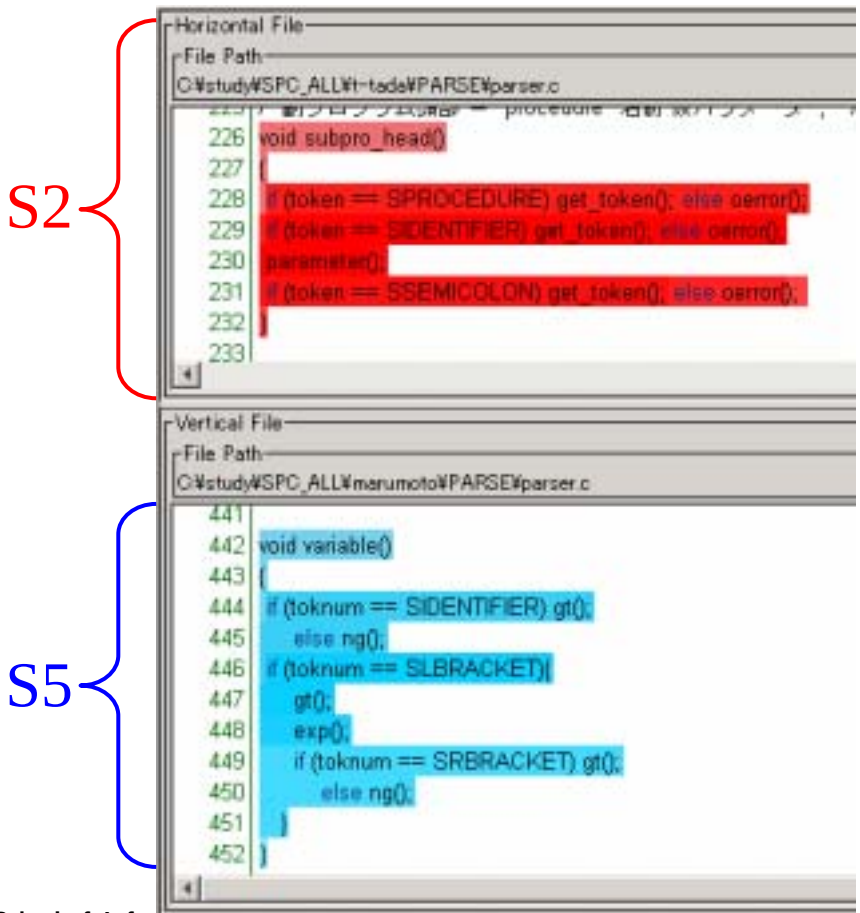
Students Excise

- 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 semantic checker
- Purpose
  - Check the stepwise development
  - Check plagiarisms

Result

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





Case study 3:

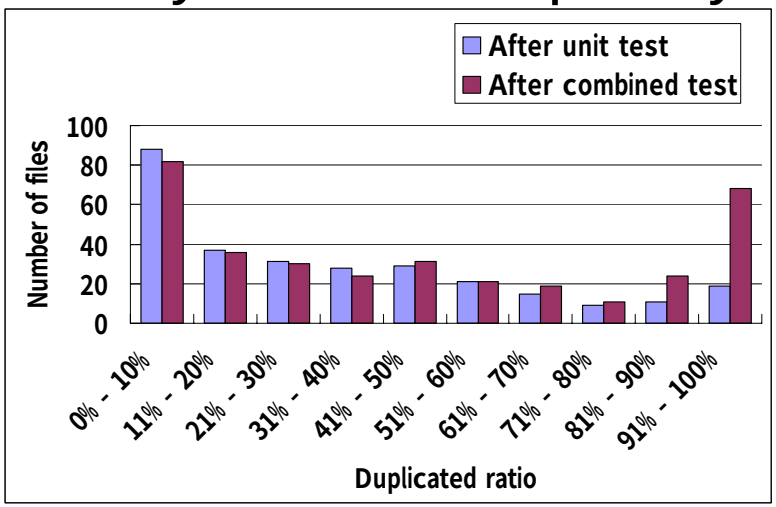
IPA/SEC Advanced Project

- 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 indelepndently 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

IPA/SEC Advanced Project:

Duplicated Ratio

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

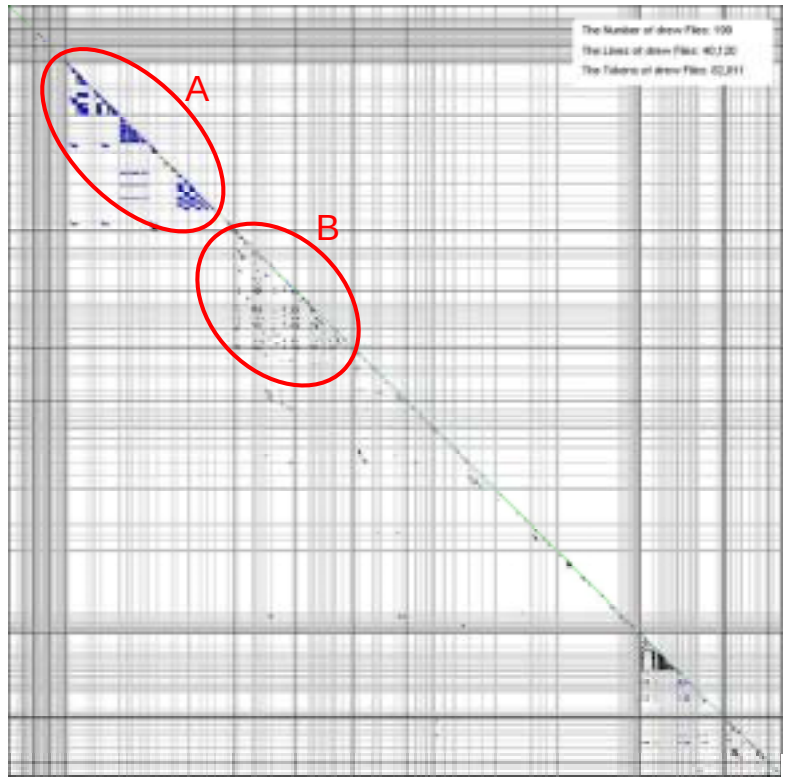


- We interviewed developers of the sub-system
  - They added library code to the system to add new functions right before combined test

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 track
  - Logical structures are mostly the same



IPA/SEC Advanced Project:

Clone Metrics Analysis

- LEN: A clone set detected from a company included 154-lines code fragments
  - A code fragment was in file AAAXXXBBB.cpp
  - The other code fragment was in file AAAYYYBBB.cpp
  - In code fragment of AAAYYYBBB.cpp, some function names and comments include XXX
    - This implies that a `copy-and-paste' was done from AAAXXXBBB.cpp to AAAYYYBBB.cpp

Clone Metrics Analysis

- NIF: The greatest value of NIF of a company was 8 (The clone set involved in 8 files)
  - Each code fragment checks whether or not the end of string is NULL. If not, add NULL
  - Whole of methods were duplicated
    - It means that these code clones are easily merged by moving to utility package

File Metrics Analysis

- NOC: A file contained 358 code clones
  - Code clones were scattered widely in the file
  - No bug-related code clones were found, but the maintainability of the file is questionable
    - The file size is very big (over 10KLines)
    - Various processes are included

File Metrics Analysis

- ROC: Two files had very high duplicated ratio(96%)
  - A file is for an off-line process
  - The other file is for an on-line process with the same algorithm
  - Developers knew the presence of these code clones
    - In the design process, they decided to separate off-line and on-line processes

Summary of Code Clone Analysis and Application

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

Future Direction

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

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

END