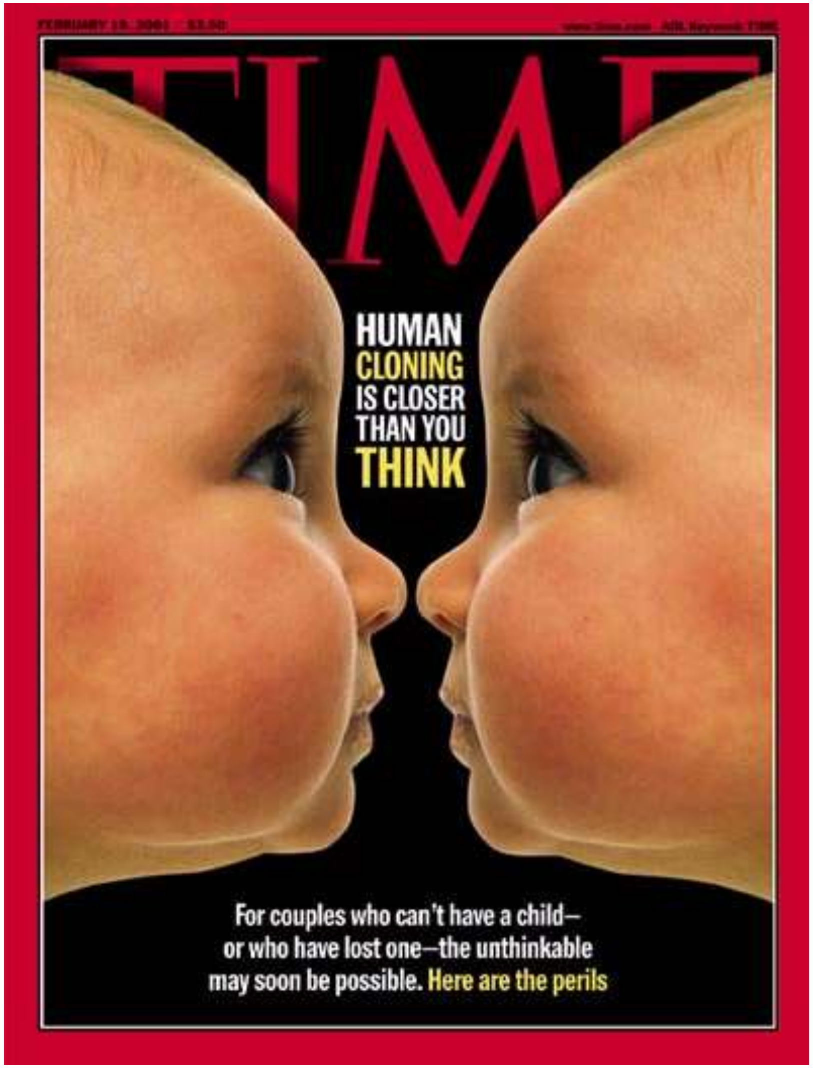
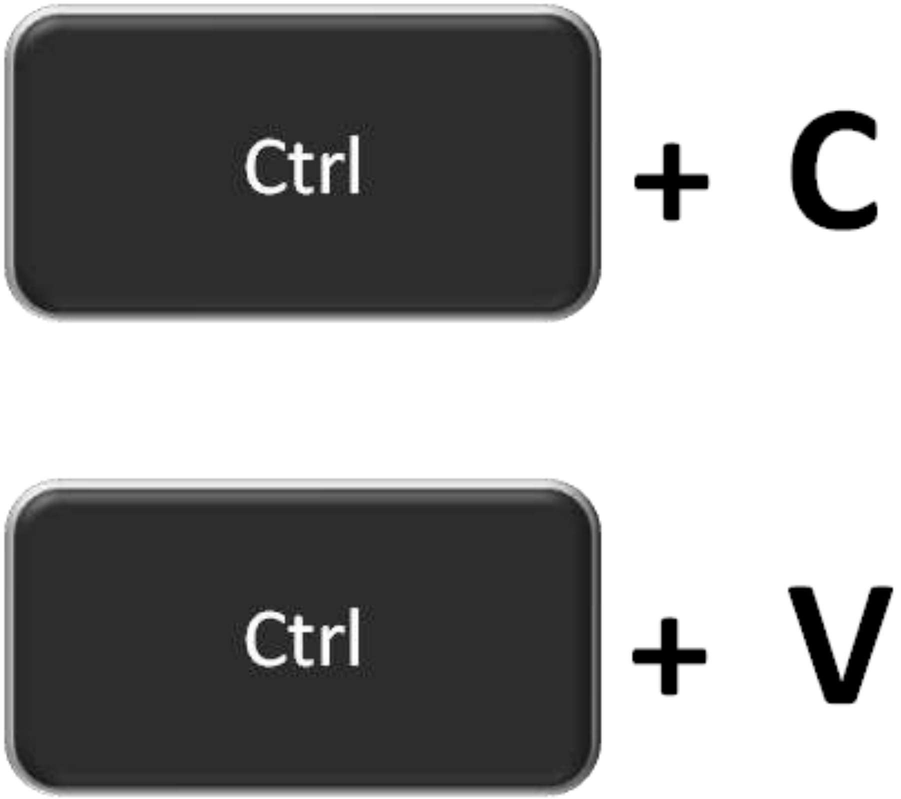

TIME
HUMAN CLONING IS CLOSER THAN YOU THINK
For couples who can't have a child— or who have lost one—the unthinkable may soon be possible. Here are the perils

Clones

Human Cloning is challenging! Right?

What about cloning in software development?

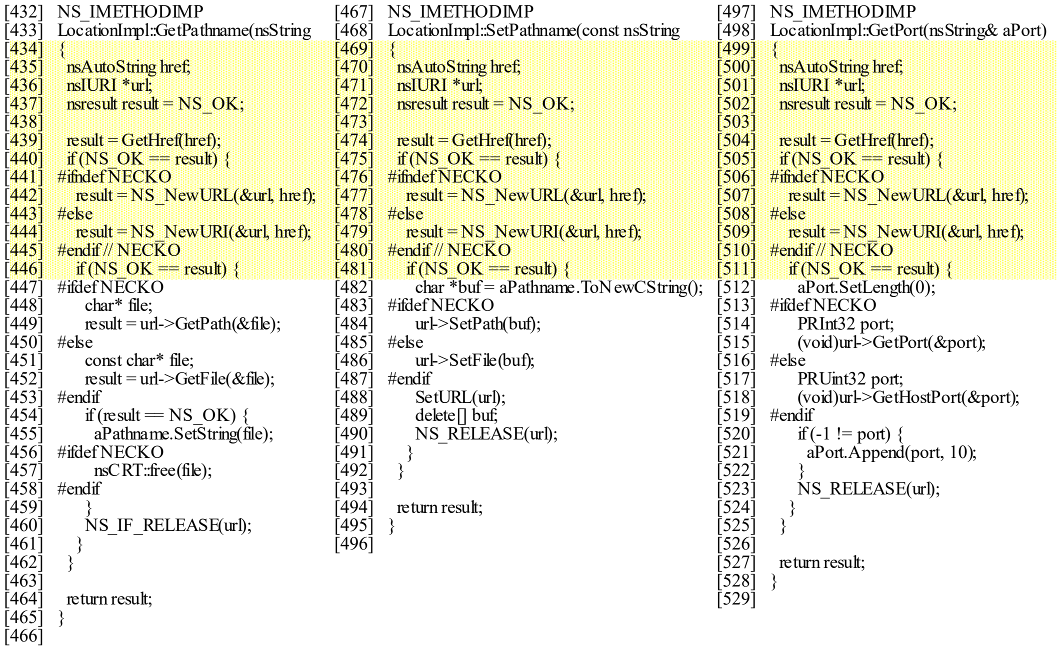
As easy as:



Code is Copied

Small Example from the Mozilla Distribution (Milestone 9)

Extract from /dom/src/base/nsLocation.cpp



```
NS_IMETHODIMP
LocationImpl::GetPathname(nsString
{
  nsAutoString href;
  nsIURI *url;
  nsresult result = NS_OK;

  result = GetHref(href);
  if (NS_OK == result) {
#ifndef NECKO
    result = NS_NewURL(&url, href);
#else
    result = NS_NewURI(&url, href);
#endif // NECKO
    if (NS_OK == result) {
#ifdef NECKO
      char* file;
      result = url->GetPath(&file);
#else
      const char* file;
      result = url->GetFile(&file);
#endif
      if (result == NS_OK) {
        aPathname.SetString(file);
#ifdef NECKO
        nsCRT::free(file);
#endif
      }
      NS_IF_RELEASE(url);
    }
  }

  return result;
}

```

```
NS_IMETHODIMP
LocationImpl::SetPathname(const nsString
{
  nsAutoString href;
  nsIURI *url;
  nsresult result = NS_OK;

  result = GetHref(href);
  if (NS_OK == result) {
#ifndef NECKO
    result = NS_NewURL(&url, href);
#else
    result = NS_NewURI(&url, href);
#endif // NECKO
    if (NS_OK == result) {
      char *buf = aPathname.ToNewCString();
#ifdef NECKO
      url->SetPath(buf);
#else
      url->SetFile(buf);
#endif
      SetURL(url);
      delete[] buf;
      NS_RELEASE(url);
    }
  }

  return result;
}

```

```
NS_IMETHODIMP
LocationImpl::GetPort(nsString& aPort)
{
  nsAutoString href;
  nsIURI *url;
  nsresult result = NS_OK;

  result = GetHref(href);
  if (NS_OK == result) {
#ifndef NECKO
    result = NS_NewURL(&url, href);
#else
    result = NS_NewURI(&url, href);
#endif // NECKO
    if (NS_OK == result) {
      aPort.SetLength(0);
#ifdef NECKO
      PRInt32 port;
      (void)url->GetPort(&port);
#else
      PRUint32 port;
      (void)url->GetHostPort(&port);
#endif
      if (-1 != port) {
        aPort.Append(port, 10);
      }
      NS_RELEASE(url);
    }
  }

  return result;
}

```

Why Clones are bad?!

General negative effect

- Code bloat

Negative effects on Software Maintenance

- Copied Defects
- Changes take double, triple, quadruple, ... Work
- Dead code
- Add to the cognitive load of future maintainers

Copying as additional source of defects

- Errors in the systematic renaming produce unintended aliasing

How Much Code is Duplicated?

Usual estimates: 8 to 12% in normal industrial code

15 to 25 % is already a lot!

Case Study	LOC	Duplication without comments	with comments
gcc	*460’000*	*8.7%*	*5.6%*
Database Server	*245’000*	*36.4%*	*23.3%*
Payroll	*40’000*	*59.3%*	*25.4%*
Message Board	*6’500*	*29.4%*	*17.4%*

What is Duplicated Code?

- Duplicated Code = Source code segments that are found in different places of a system.
  - in different files
  - in the same file but in different functions
  - in the same function
- The segments must contain some logic or structure that can be abstracted, i.e.,

```
...
computeIt(a,b,c,d);
...
```

```
...
computeIt(w,x,y,z);
...
```

is not considered duplicated code.

```
...
getIt(hash(tail(z)));
...
```

```
...
getIt(hash(tail(a)));
...
```

could be abstracted to a new function

- Copied artifacts range from expressions, to functions, to data structures, and to entire subsystems.

Why Cloning Occurs

Development Time

- Cloning a procedure rather than extracting a common part may save on time

Communication

- A code set may be borrowed but its working might not be clear

Structural

- Code borrowed from an un-modifiable subsystem.

Coincidence

- Look Alikes and clones are difficult to differentiate.

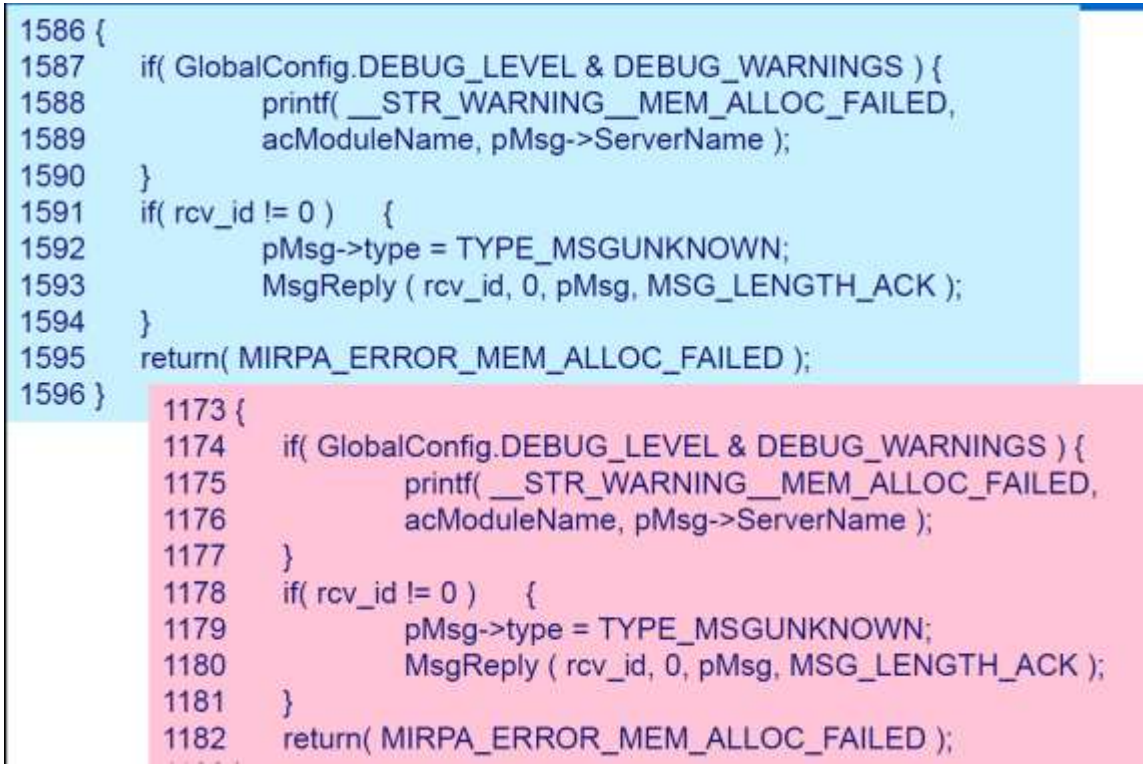
Definitions

Clone Class/Set: Set of equivalent Clones

Clone-pair

Precision: Percent of reported clones that are genuine

Recall: Percent of genuine clones that are reported

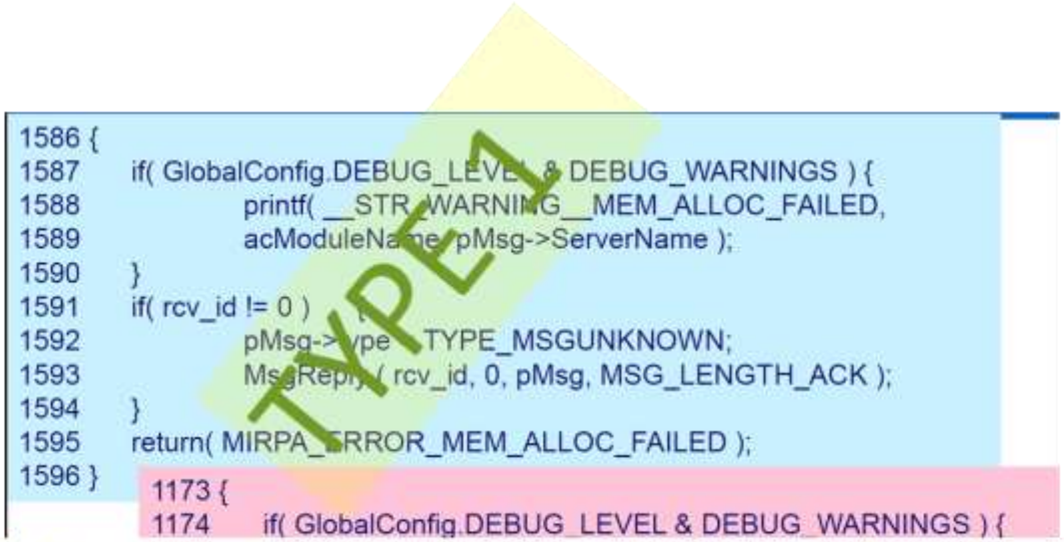


```
{
    if( GlobalConfig.DEBUG_LEVEL & DEBUG_WARNINGS ) {
            printf( __STR_WARNING__MEM_ALLOC_FAILED,
            acModuleName, pMsg->ServerName );
    }
    if( rcv_id != 0 )    {
            pMsg->type = TYPE_MSGUNKNOWN;
            MsgReply ( rcv_id, 0, pMsg, MSG_LENGTH_ACK );
    }
    return( MIRPA_ERROR_MEM_ALLOC_FAILED );
}
```

```
{
    if( GlobalConfig.DEBUG_LEVEL & DEBUG_WARNINGS ) {
            printf( __STR_WARNING__MEM_ALLOC_FAILED,
            acModuleName, pMsg->ServerName );
    }
    if( rcv_id != 0 )    {
            pMsg->type = TYPE_MSGUNKNOWN;
            MsgReply ( rcv_id, 0, pMsg, MSG_LENGTH_ACK );
    }
    return( MIRPA_ERROR_MEM_ALLOC_FAILED );
```

Clones?

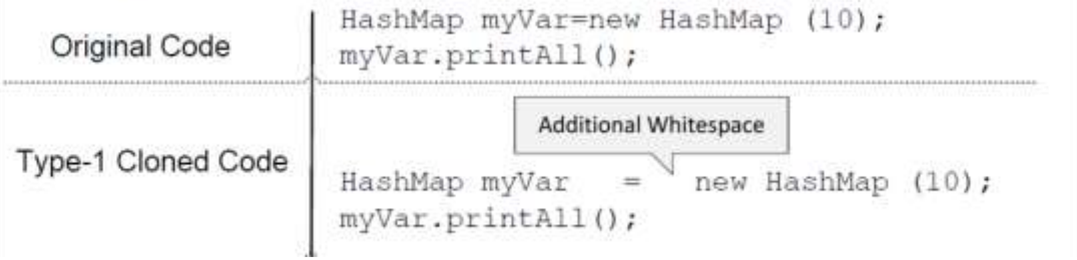
Type 1 Clones



```
{
    if( GlobalConfig.DEBUG_LEVEL & DEBUG_WARNINGS ) {
            printf( __STR_WARNING__MEM_ALLOC_FAILED,
            acModuleName, pMsg->ServerName );
    }
    if( rcv_id != 0 ) {
            pMsg->type = TYPE_MSGUNKNOWN;
            MsgReply( rcv_id, 0, pMsg, MSG_LENGTH_ACK );
    }
    return( MIRPA_ERROR_MEM_ALLOC_FAILED );
}
```

```
{
    if( GlobalConfig.DEBUG_LEVEL & DEBUG_WARNINGS ) {
```

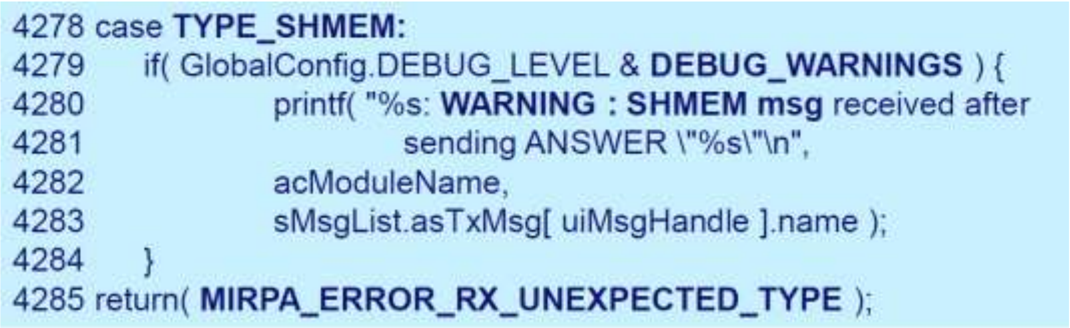
Type 1: They are identical up to whitespace/comments


Additional Whitespace

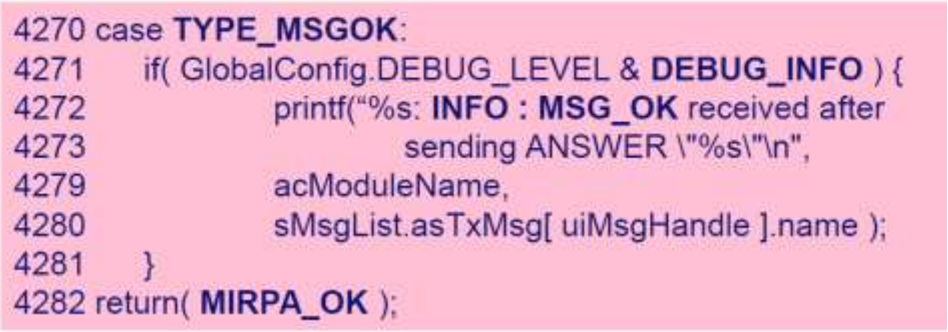
Original Code	`HashMap myVar=new HashMap (10);`
	`myVar.printAll();`

Type-1 Cloned Code	`HashMap myVar    =   new HashMap (10);`
	`myVar.printAll();`

Clones?

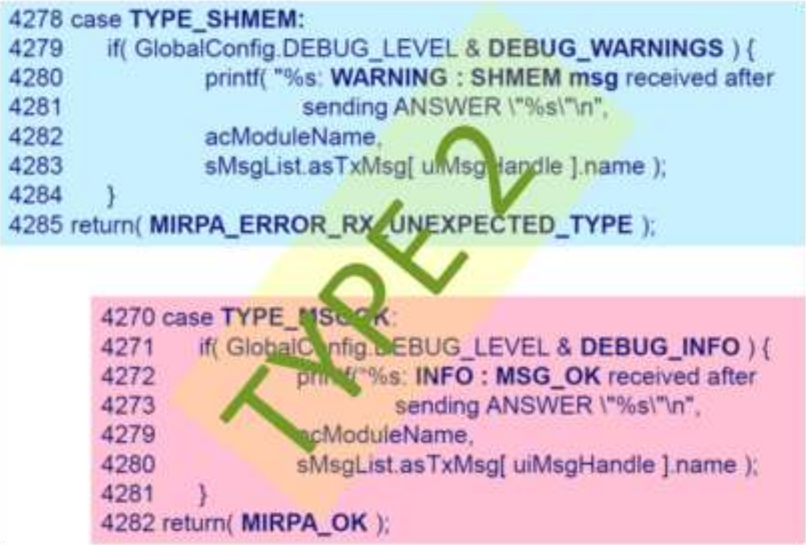


```
case TYPE_SHMEM:
    if( GlobalConfig.DEBUG_LEVEL & DEBUG_WARNINGS ) {
            printf( "%s: WARNING : SHMEM msg received after
                    sending ANSWER \"%s\"\n",
            acModuleName,
            sMsgList.asTxMsg[ uiMsgHandle ].name );
    }
return( MIRPA_ERROR_RX_UNEXPECTED_TYPE );
```



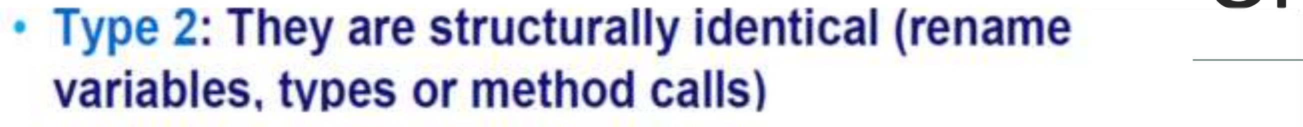
```
case TYPE_MSGOK:
    if( GlobalConfig.DEBUG_LEVEL & DEBUG_INFO ) {
            printf("%s: INFO : MSG_OK received after
                    sending ANSWER \"%s\"\n",
            acModuleName,
            sMsgList.asTxMsg[ uiMsgHandle ].name );
    }
return( MIRPA_OK );
```

Type 2 clones

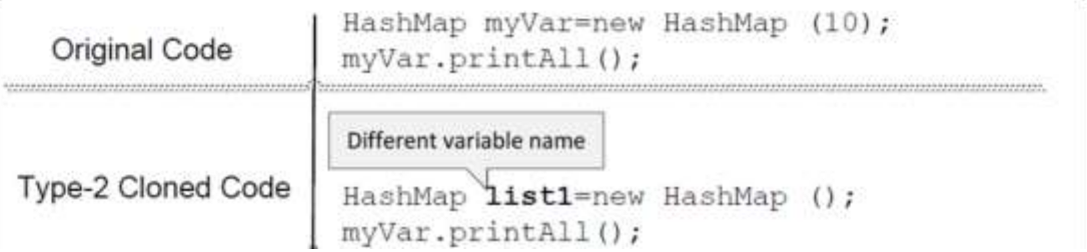


```
case TYPE_SHMEM:
    if( GlobalConfig.DEBUG_LEVEL & DEBUG_WARNINGS ) {
            printf( "%s: WARNING : SHMEM msg received after
                    sending ANSWER \"%s\"\n",
            acModuleName,
            sMsgList.asTxMsg[ uiMsgHandle ].name );
    }
return( MIRPA_ERROR_RX_UNEXPECTED_TYPE );
```

```
case TYPE_MSG_OK:
    if( GlobalConfig.DEBUG_LEVEL & DEBUG_INFO ) {
            printf( "%s: INFO : MSG_OK received after
                    sending ANSWER \"%s\"\n",
            acModuleName,
            sMsgList.asTxMsg[ uiMsgHandle ].name );
    }
return( MIRPA_OK );
```



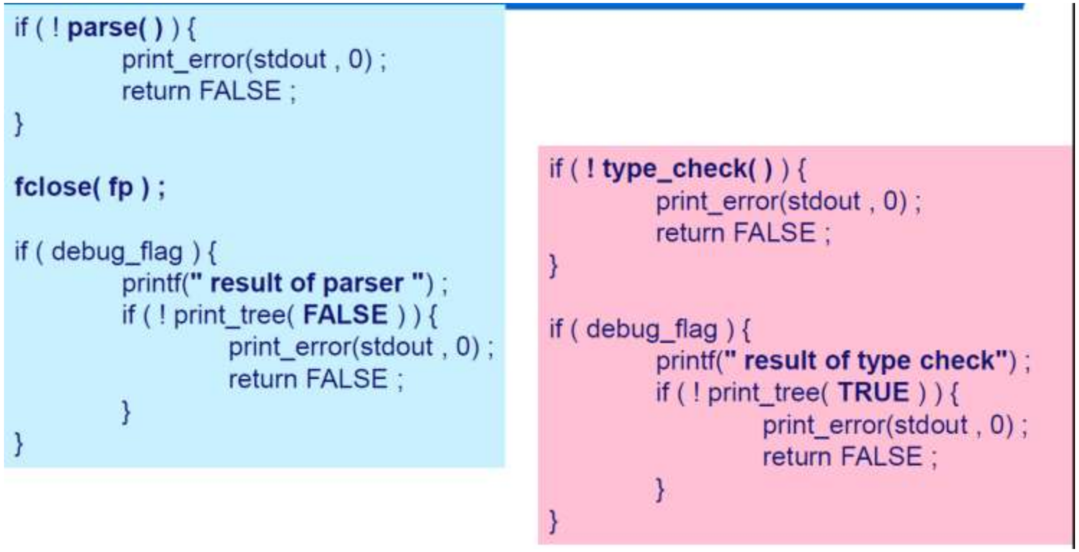
- **Type 2: They are structurally identical (rename variables, types or method calls)**



Original Code	`HashMap myVar=new HashMap (10);`
	`myVar.printAll();`

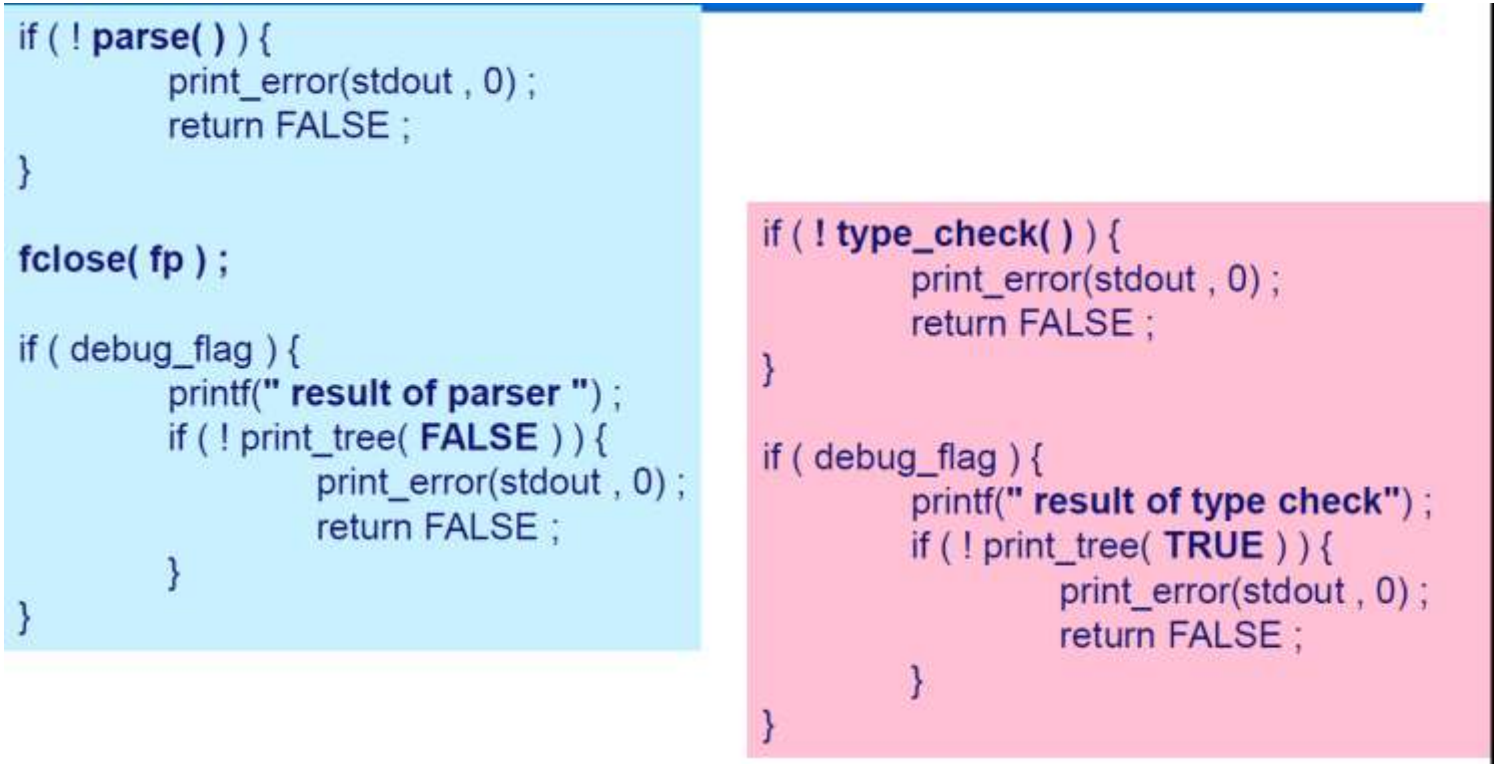
Type-2 Cloned Code	Different variable name
	`HashMap list1=new HashMap ();`
	`myVar.printAll();`

Clones?



```
if ( ! parse( ) ) {
        print_error(stdout , 0) ;
        return FALSE ;
}

fclose( fp ) ;

if ( debug_flag ) {
        printf(" result of parser ") ;
        if ( ! print_tree( FALSE ) ) {
                print_error(stdout , 0) ;
                return FALSE ;
        }
}
```

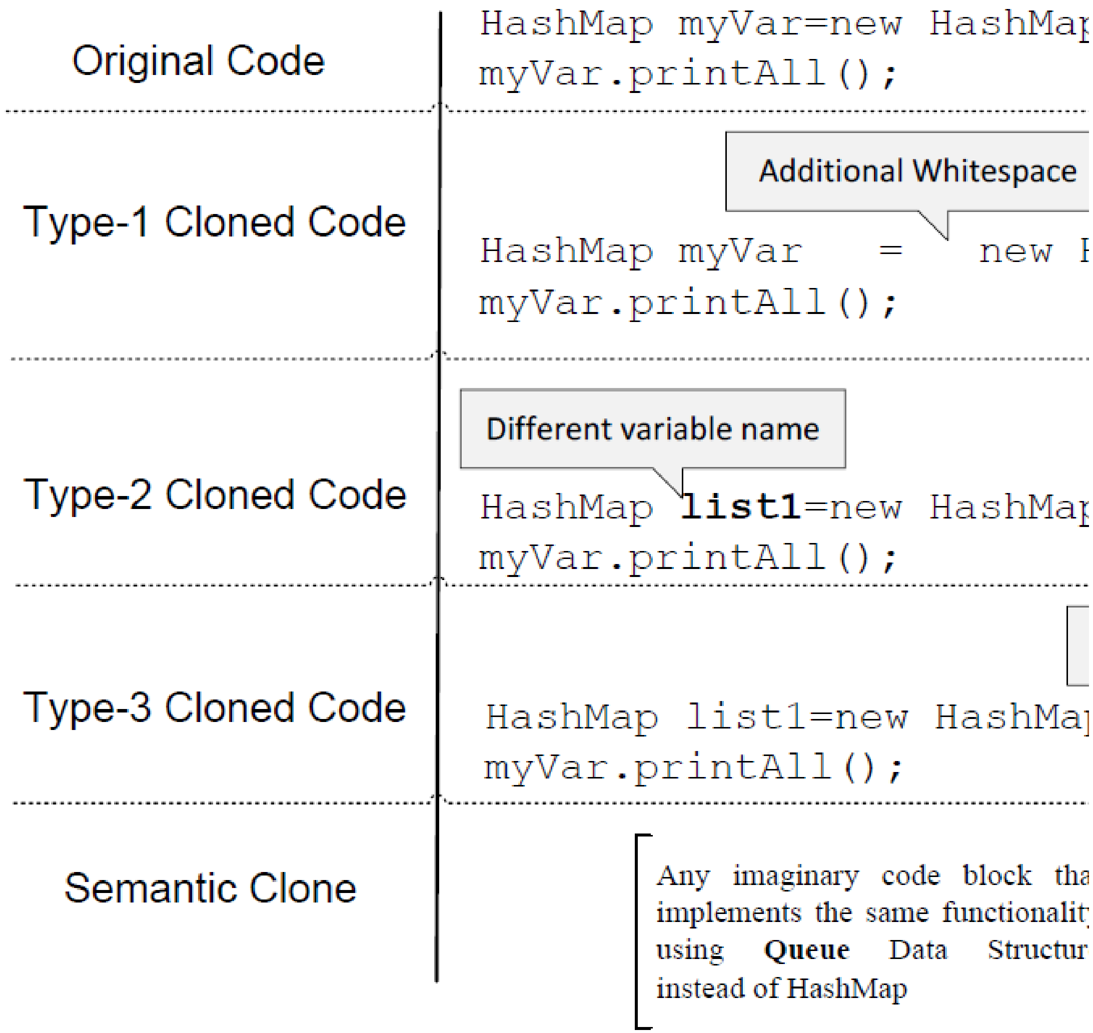
```
if ( ! type_check( ) ) {
        print_error(stdout , 0) ;
        return FALSE ;
}

if ( debug_flag ) {
        printf(" result of type check") ;
        if ( ! print_tree( TRUE ) ) {
                print_error(stdout , 0) ;
                return FALSE ;
        }
}
```



```
if ( ! parse( ) ) {
            print_error(stdout , 0) ;
            return FALSE ;
}

fclose( fp ) ;

if ( debug_flag ) {
            printf(" result of parser ") ;
            if ( ! print_tree( FALSE ) ) {
                        print_error(stdout , 0) ;
                        return FALSE ;
            }
}
```

```
if ( ! type_check( ) ) {
            print_error(stdout , 0) ;
            return FALSE ;
}

if ( debug_flag ) {
            printf(" result of type check") ;
            if ( ! print_tree( TRUE ) ) {
                        print_error(stdout , 0) ;
                        return FALSE ;
            }
}
```

Type 3 Clones



Original Code

```
HashMap myVar=new HashMa
myVar.printAll();
```

Type-1 Cloned Code

```
HashMap myVar    =    new
myVar.printAll();
```

Type-2 Cloned Code

```
HashMap list1=new HashMa
myVar.printAll();
```

Type-3 Cloned Code

```
HashMap list1=new HashMa
myVar.printAll();
```

Semantic Clone

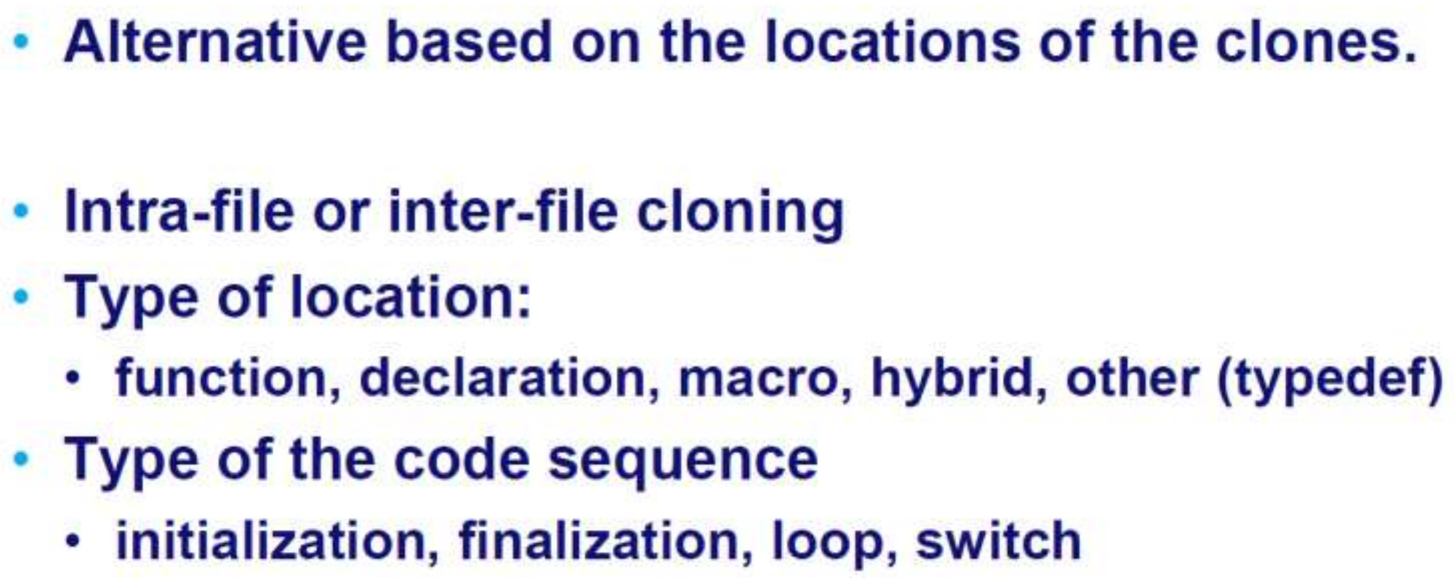
Any imaginary code block tha implements the same functionalit using **Queue** Data Structur instead of HashMap

Clone Definition (Source Code Clone)

Similar code fragments

- Type 1: Identical except whitespaces …
- Type 2: Identical except variable names ...
- Type 3: Identical except a few missing…
- Type 4: Similar functionality

Alternative Classification



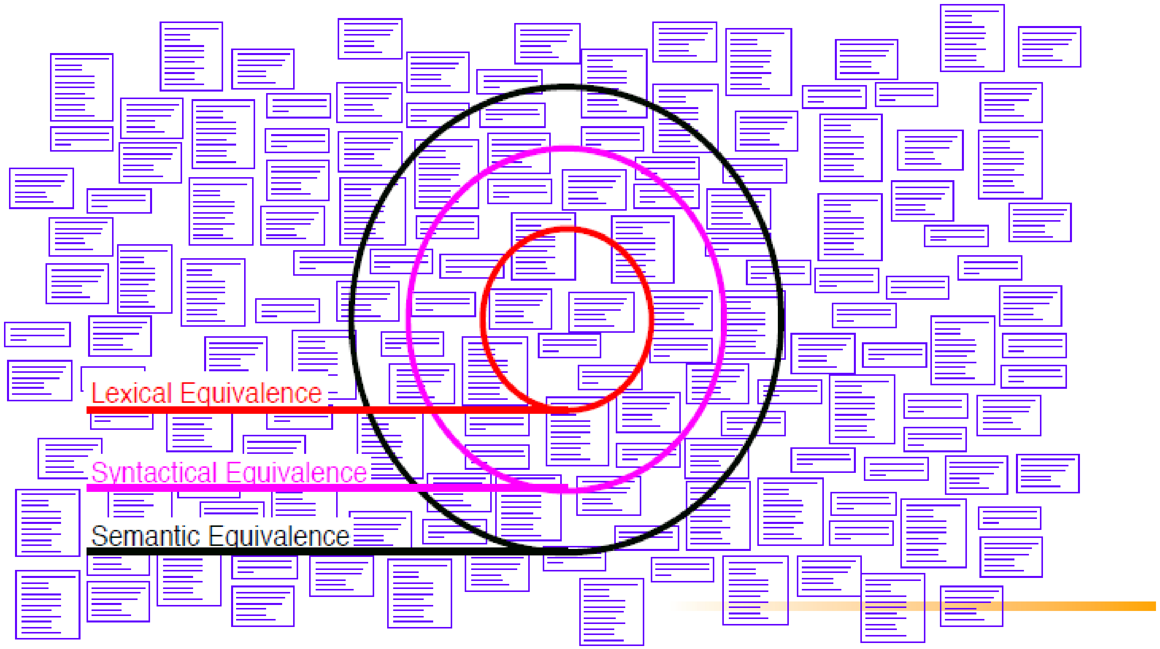
- Alternative based on the locations of the clones.

- Intra-file or inter-file cloning
- Type of location:
  - function, declaration, macro, hybrid, other (typedef)
- Type of the code sequence
  - initialization, finalization, loop, switch

Code Duplication Detection

Nontrivial problem:

- No a priori knowledge about which code has been copied
- How to find all clone pairs among all possible pairs of segments?



Code Detection

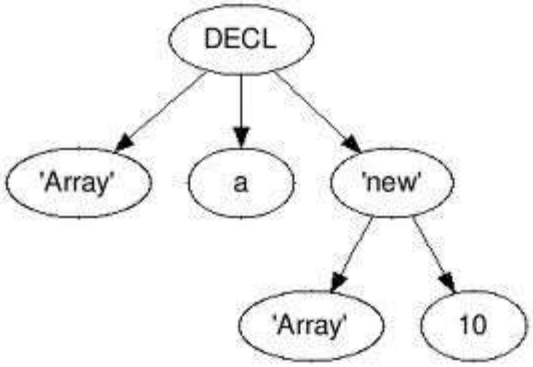
Detection Techniques

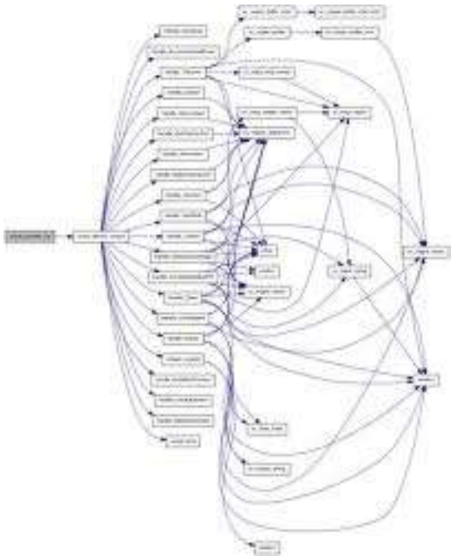
String Matching – Represents and evaluates code using string comparisons.

Token Parsing – Code transformation into tokens for comparison.

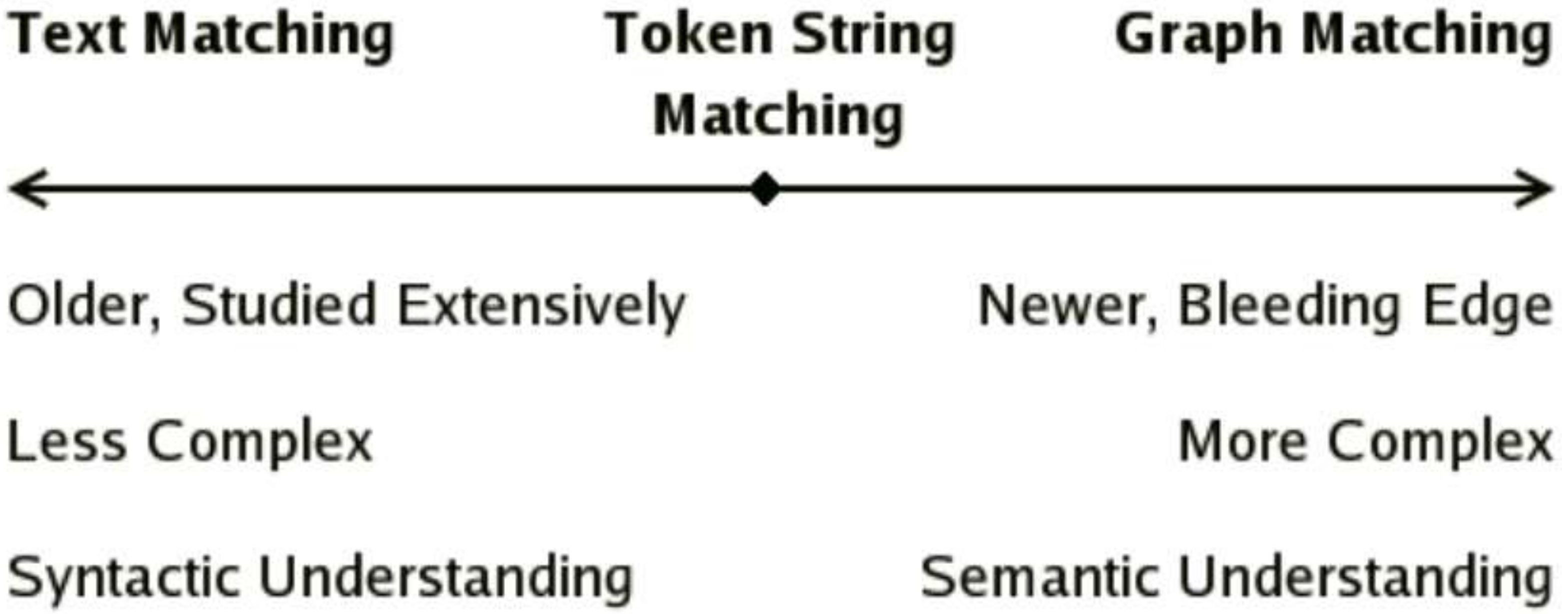
Graph Matching – Pattern matching on graph representations of code.

```
Array a = new Array(10);
```





Detection Strategies



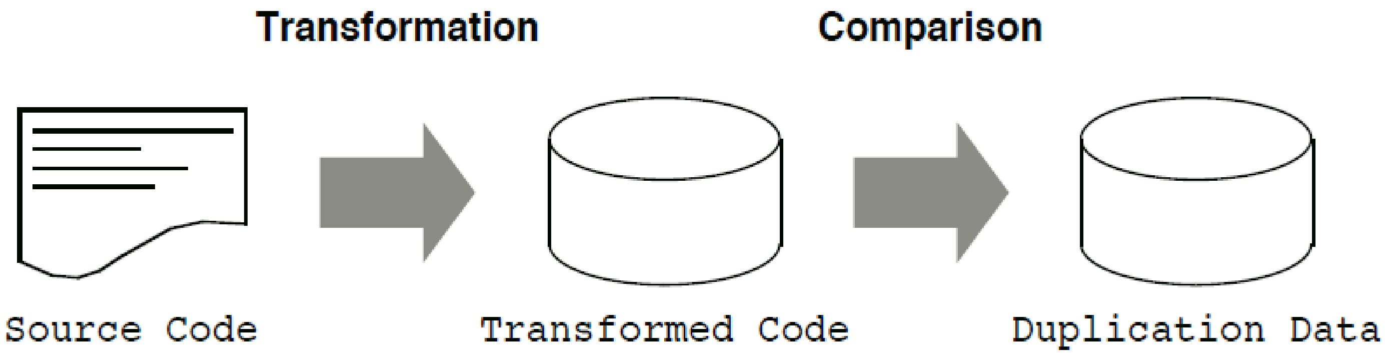
Text Matching | **Token String Matching** | **Graph Matching**

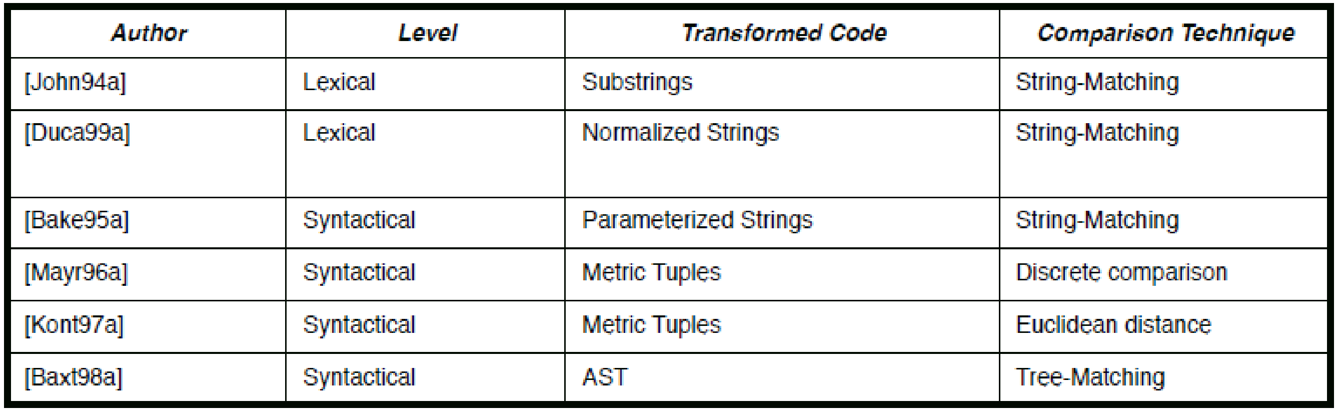
Older, Studied Extensively | Newer, Bleeding Edge

Less Complex | More Complex

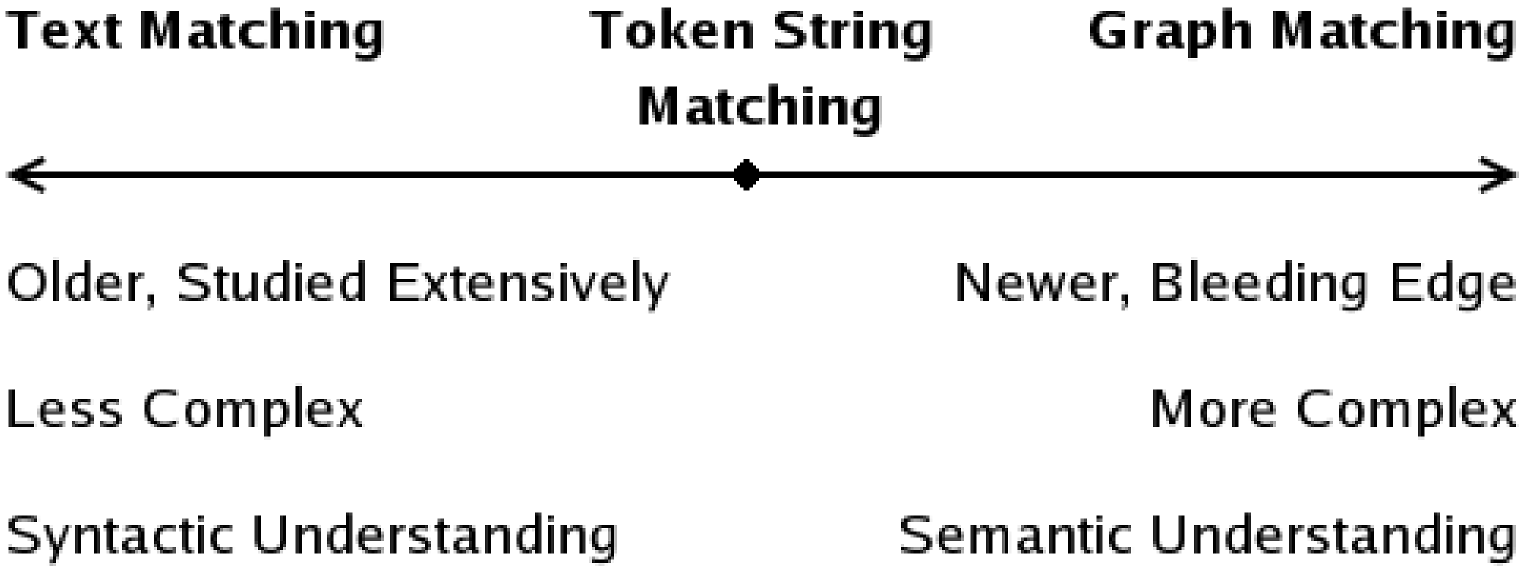
Syntactic Understanding | Semantic Understanding

General Schema of Detection Process





Author	*Level*	*Transformed Code*	*Comparison Technique*
[John94a]	Lexical	Substrings	String-Matching
[Duca99a]	Lexical	Normalized Strings	String-Matching
[Bake95a]	Syntactical	Parameterized Strings	String-Matching
[Mayr96a]	Syntactical	Metric Tuples	Discrete comparison
[Kont97a]	Syntactical	Metric Tuples	Euclidean distance
[Baxt98a]	Syntactical	AST	Tree-Matching



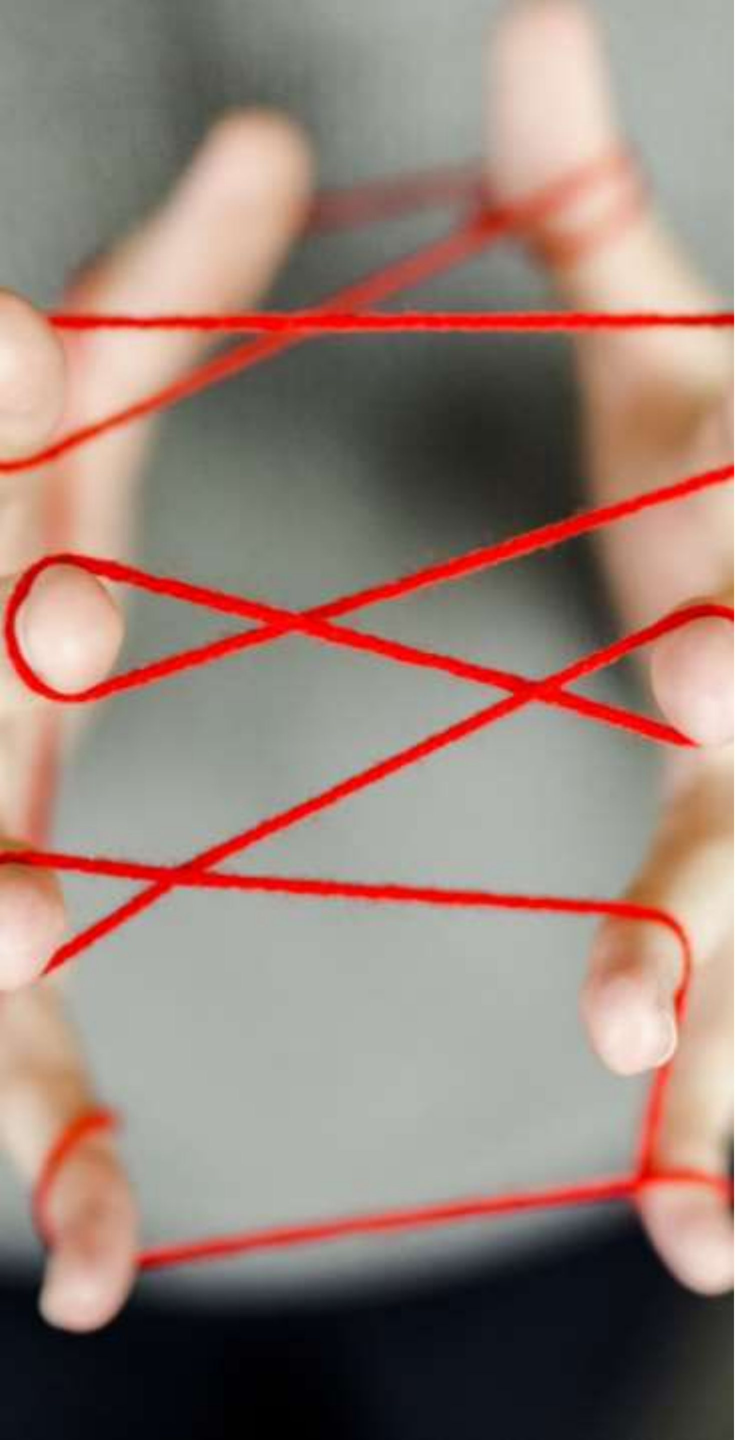
Text Matching | **Token String Matching** | **Graph Matching**

Older, Studied Extensively | Newer, Bleeding Edge

Less Complex | More Complex

Syntactic Understanding | Semantic Understanding

Detection Strategies



Exact String Matching

Definition

- Two sections of code are said to be a maximal exact match if their lines match exactly character by character but the preceding lines do not match and the following lines do not match.

Parameterized String Example

□ Was found in the X-Window C code

□ Fragment 1:

```
copy-number (&pmin, &pmax ,
pfi->min-bounds.lbearing,
pfi->max-bounds.lbeaing);
*pmin++ = *pmax++ = J , J ;
copy-number(&pmin, kpmax,
pfi->min-bounds.rbearing,
pf i->max-bounds .rbearing) ;
*pmin++ = *pmax++ = J , J ;
```

□ Fragment 2:

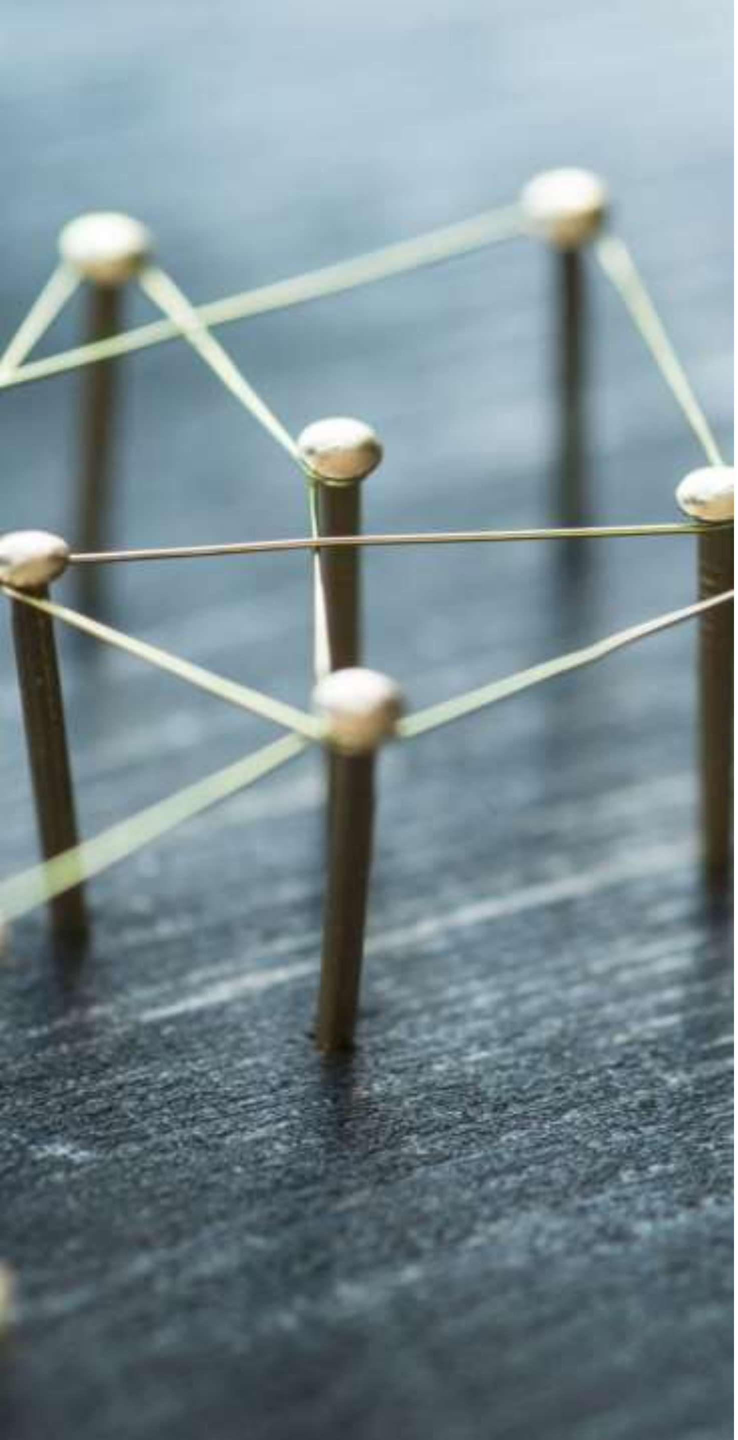
```
copy-number(&pmin, &pmax,
pfh->min-bounds.left,
pfh->max-bounds.left);
*pmin++ = *pmax++ = J , J ;
copy-number(&pmin, &pmax,
pfh->min-bounds.right,
pfh->max-bounds.right);
*pmin++ = *pmax++ = J , J ;
```

Substring Matching

Substring Matching provides a faster search algorithm.

Phases

1. Normalization
2. Substring Generation
3. Matching
4. Consolidation
5. Reporting

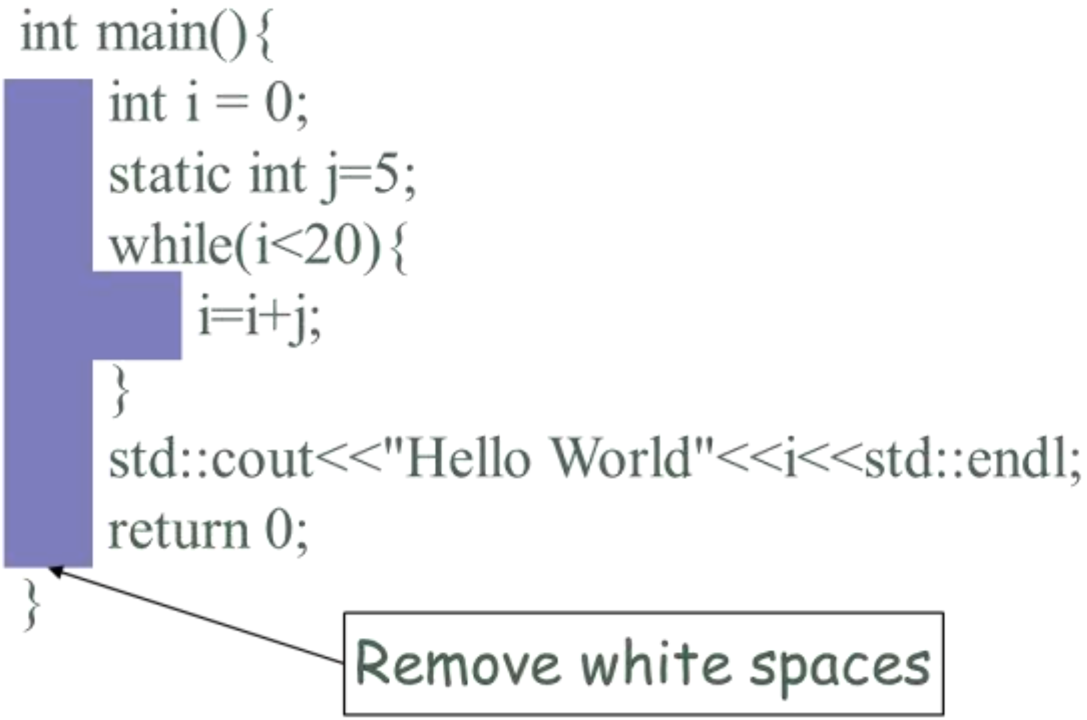


Token Parsing Techniques

Transforms code into tokens by using language specific constructs

Find similarities within this token string

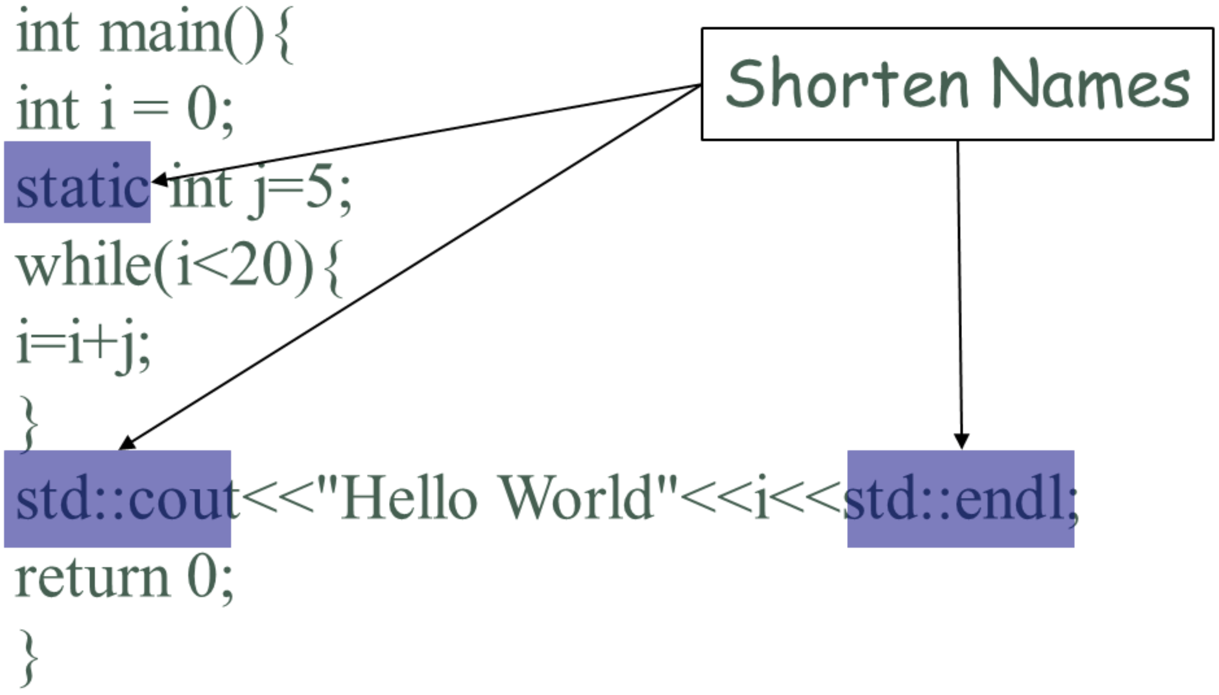
Transform token clones back into code clones for presentation



```
int main(){
    int i = 0;
    static int j=5;
    while(i<20){
        i=i+j;
    }
    std::cout<<"Hello World"<<i<<std::endl;
    return 0;
}
```

Remove white spaces

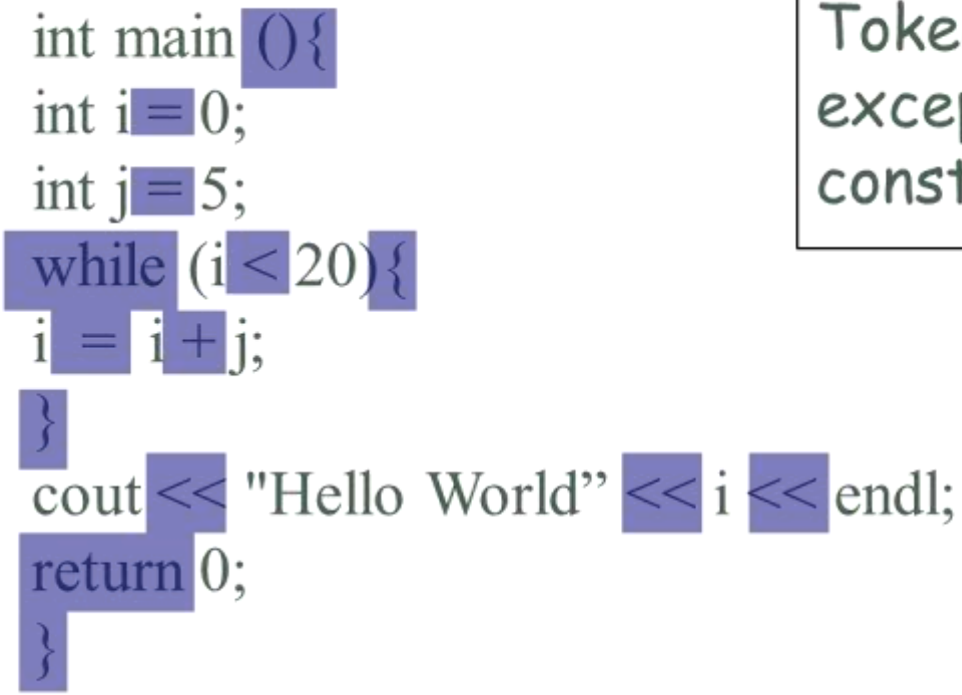
Token Parsing Example



```
int main(){
int i = 0;
static int j=5;
while(i<20){
i=i+j;
}
std::cout<<"Hello World"<<i<<std::endl;
return 0;
}
```

Shorten Names

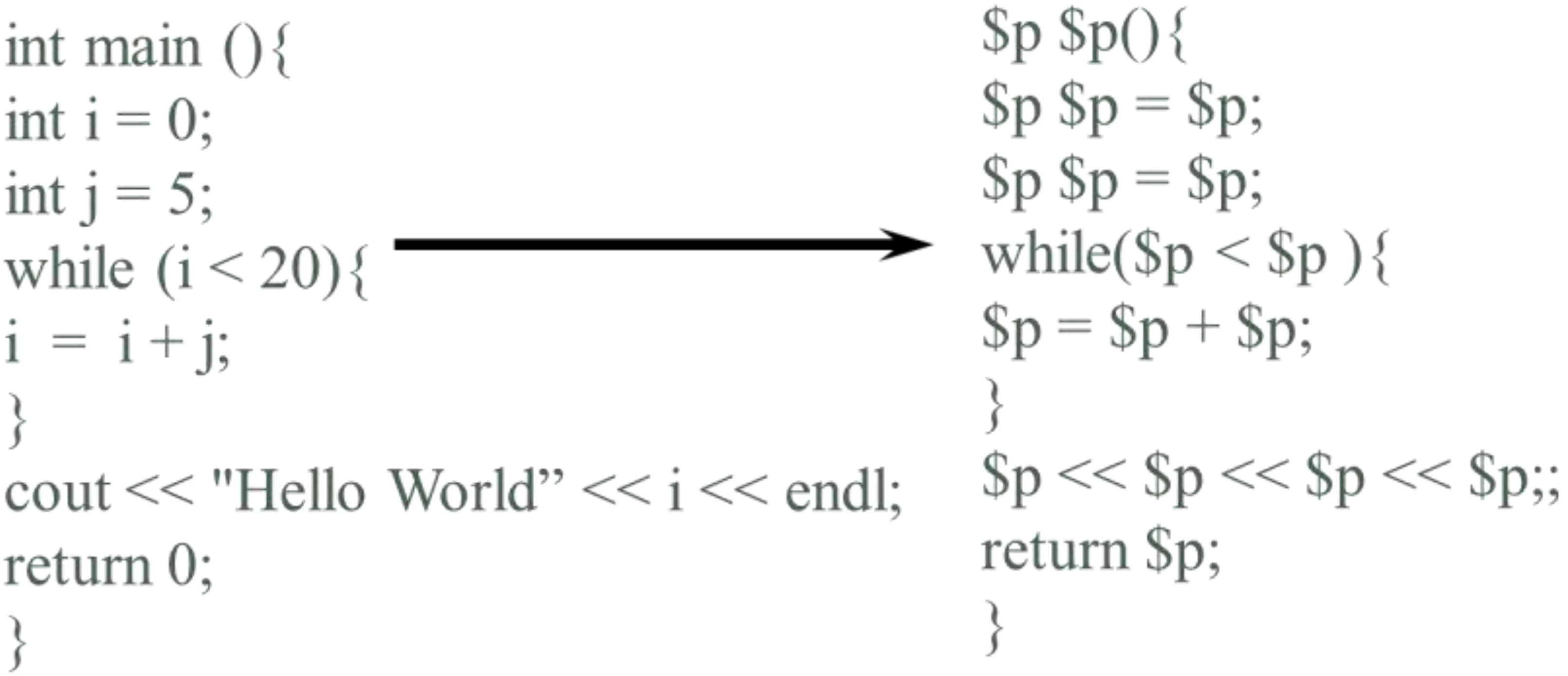
Token Parsing Example



```
int main (){
int i = 0;
int j = 5;
while (i < 20){
i = i + j;
}
cout << "Hello World” << i << endl;
return 0;
}
```

Tokenize everything, except language constructs

Token Parsing Example



```
int main (){
int i = 0;
int j = 5;
while (i < 20){
i  =  i + j;
}
cout << "Hello  World” << i << endl;
return 0;
}
```

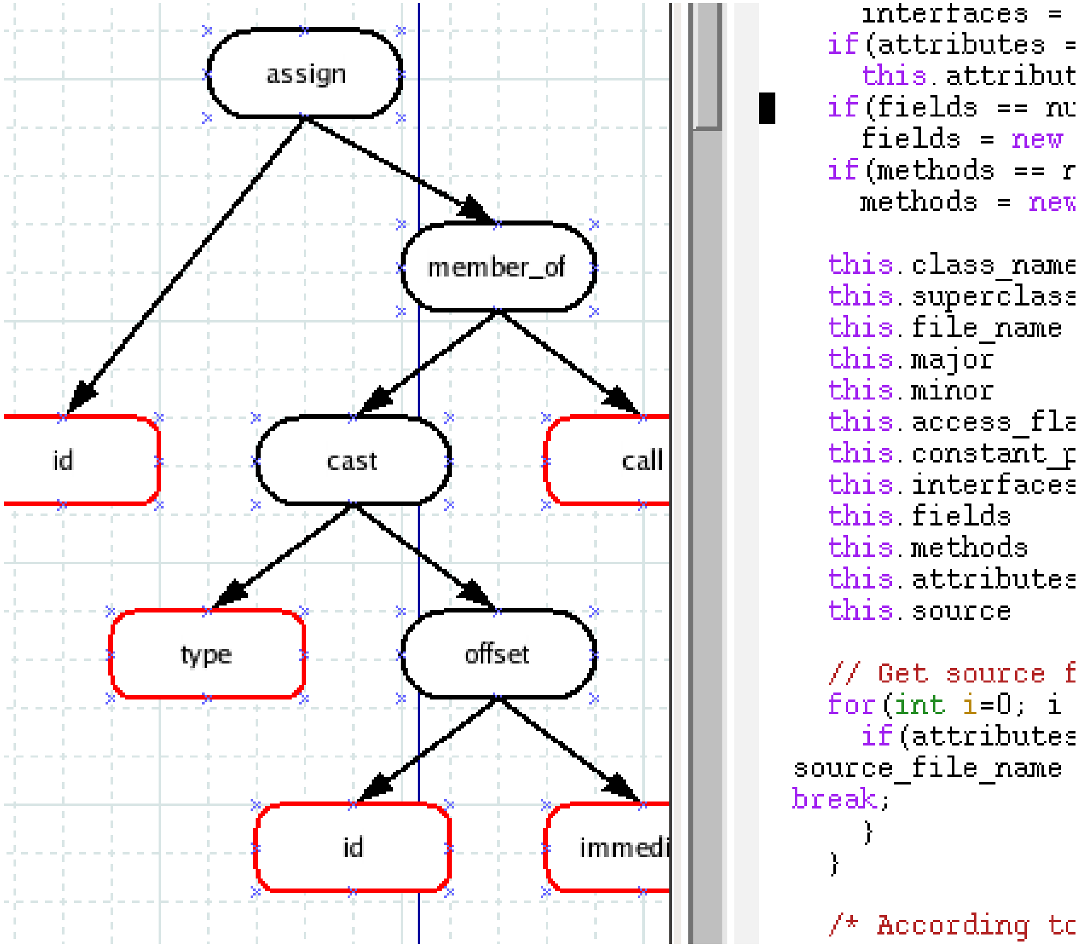
```
$p $p(){
$p $p = $p;
$p $p = $p;
while($p < $p ){
$p = $p + $p;
}
$p << $p << $p << $p;;
return $p;
}
```

Token Parsing Example

Graph Matching Techniques

- Form machine representation of code

- Identify clones as identical subgraphs



```
    interfaces =
  if(attributes =
    this.attribut
  if(fields == nu
    fields = new
  if(methods == r
    methods = new

  this.class_name
  this.superclass
  this.file_name
  this.major
  this.minor
  this.access_fla
  this.constant_p
  this.interfaces
  this.fields
  this.methods
  this.attributes
  this.source

  // Get source f
  for(int i=0; i
    if(attributes
source_file_name
break;
    }
  }

  /* According to
```

Abstract Syntax Tree Matching

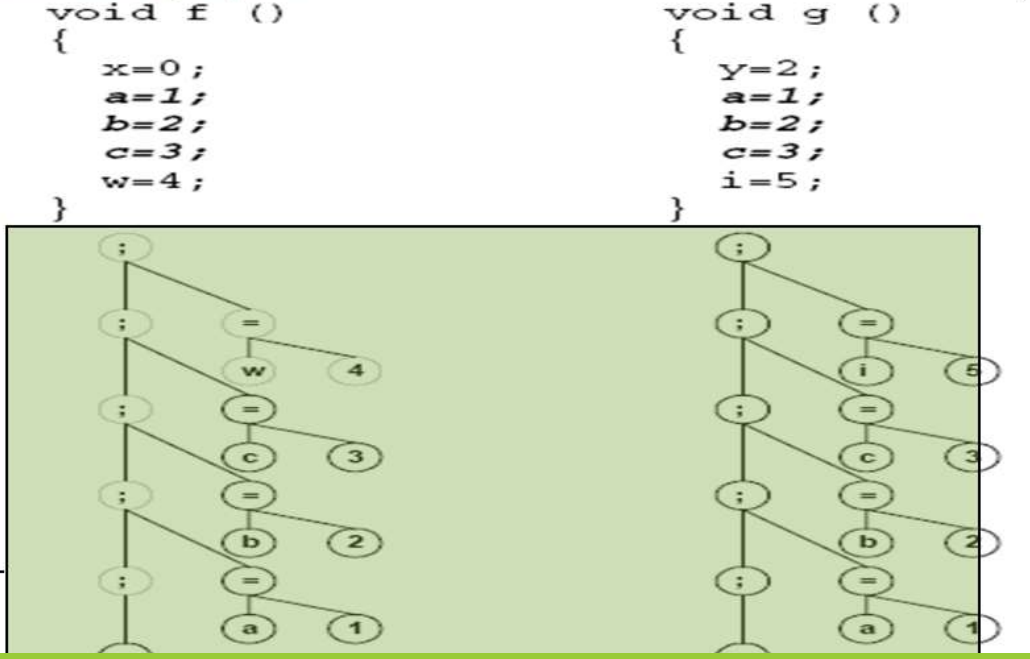
Parse Source code to build an AST

Compare subtrees by characterization metrics

Hash subgraphs

Identify maximal identical or similar subgraphs

Identify sequences of subgraphs



```
void f ()
{
   x=0;
   a=1;
   b=2;
   c=3;
   w=4;
}
```

```
void g ()
{
   y=2;
   a=1;
   b=2;
   c=3;
   i=5;
}
```

Program Dependence Graph Matching

Vertices are lines of code

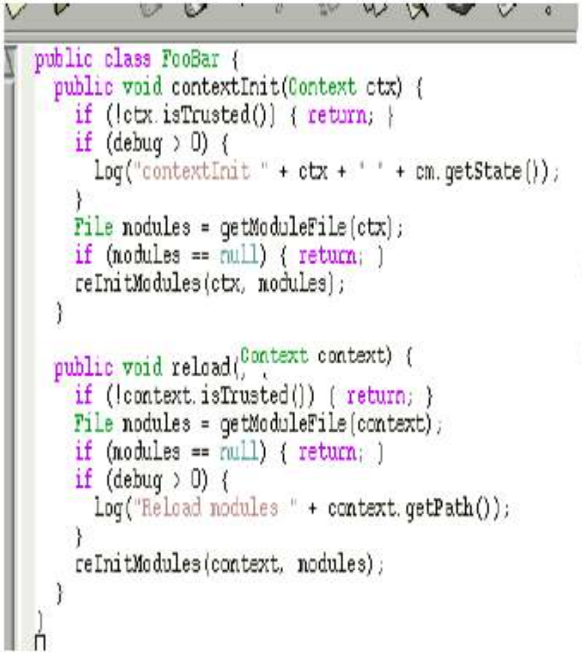
Edges are attributed with different types of dependences (control flow, data, etc)

NP complete in general, k-cutoff in maximal graph size used to limit runtime

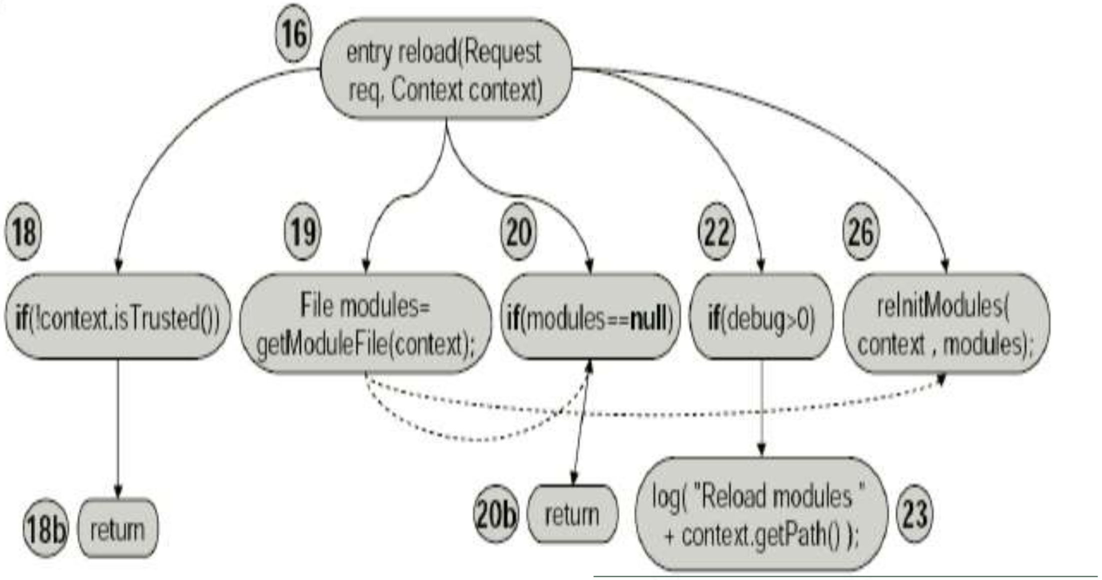
Experiments determine k=20 best

$O(|V|^2)$ possible graph starting points, reduced via heuristic

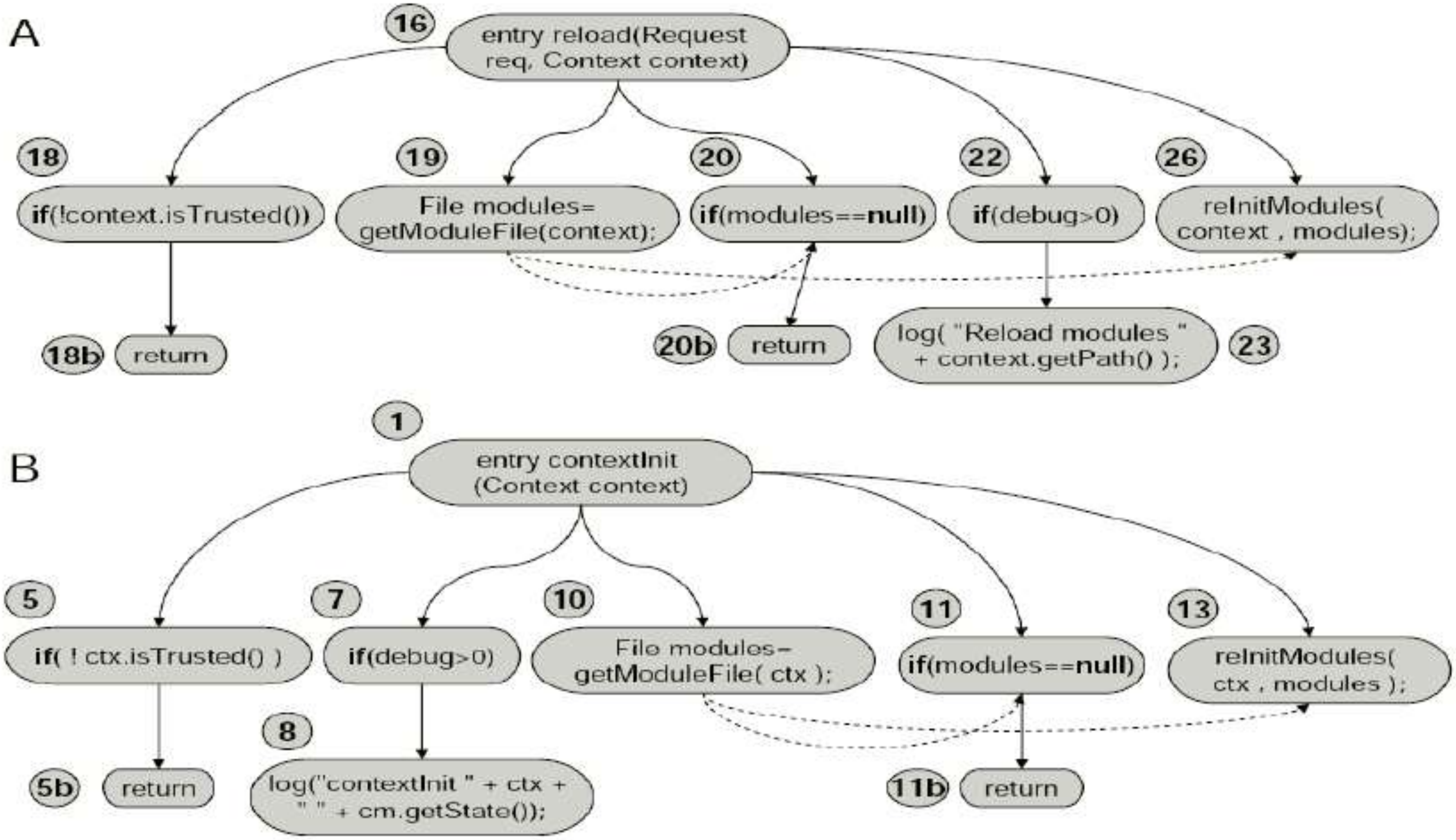
Program Dependency Graph Matching



```
public class FooBar {
  public void contextInit(Context ctx) {
    if (!ctx.isTrusted()) { return; }
    if (debug > 0) {
      Log("contextInit " + ctx + ' ' + cm.getState());
    }
    File modules = getModuleFile(ctx);
    if (modules == null) { return; }
    reInitModules(ctx, modules);
  }

  public void reload(Context context) {
    if (!context.isTrusted()) { return; }
    File modules = getModuleFile(context);
    if (modules == null) { return; }
    if (debug > 0) {
      Log("Reload modules " + context.getPath());
    }
    reInitModules(context, modules);
  }
}
```


16
entry reload(Request req, Context context)
18
if(!context.isTrusted())
19
File modules= getModuleFile(context);
20
if(modules==null)
22
if(debug>0)
26
reInitModules(context , modules);
18b
return
20b
return
log("Reload modules " + context.getPath());
23

Two Clones Found by fg-PDG


A
16
entry reload(Request req, Context context)
18
if(!context.isTrusted())
19
File modules= getModuleFile(context);
20
if(modules==null)
22
if(debug>0)
26
reInitModules(context , modules);
18b
return
20b
return
log("Reload modules " + context.getPath());
23
B
1
entry contextInit (Context context)
5
if(! ctx.isTrusted())
7
if(debug>0)
10
File modules= getModuleFile(ctx);
11
if(modules==null)
13
reInitModules(ctx , modules);
8
5b
return
log("contextInit " + ctx + " " + cm.getState());
11b
return

Metrics?

Need to evaluate different clone detection techniques

Hard to know the real number of clones in a non-trivial application

How to compare different types of clones?

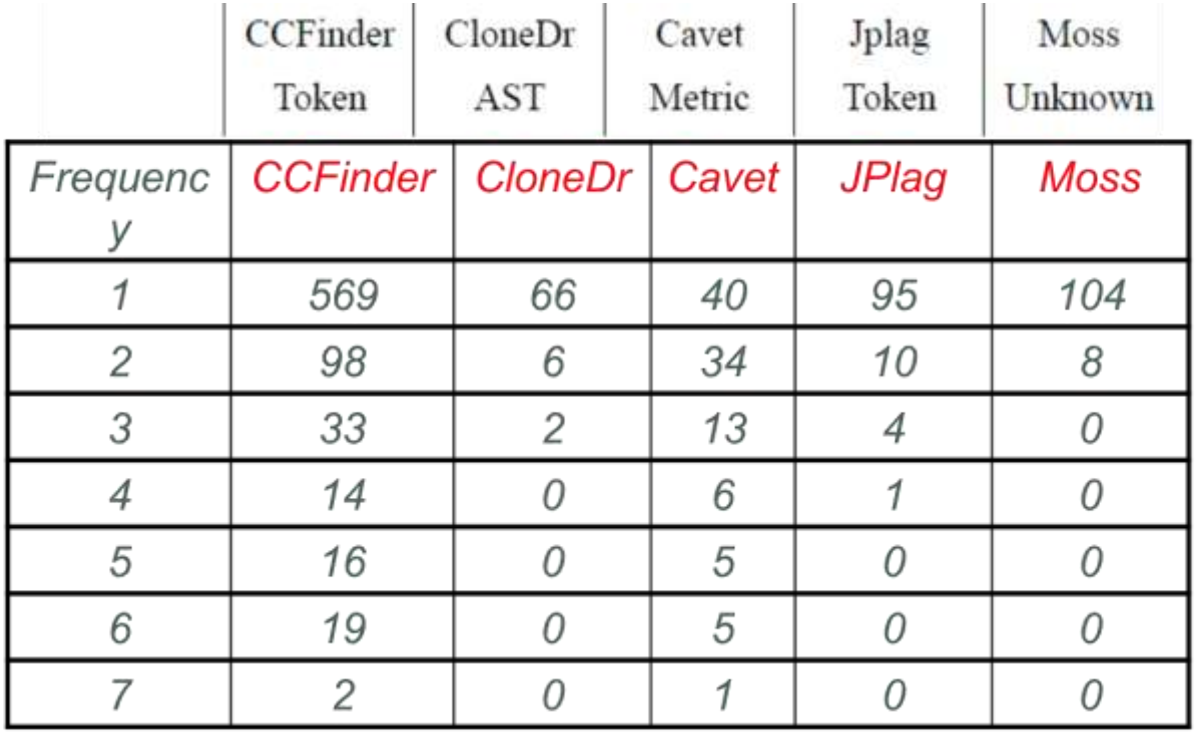
LOC: Line number count

SLOC: Line number count after the removal of blanks

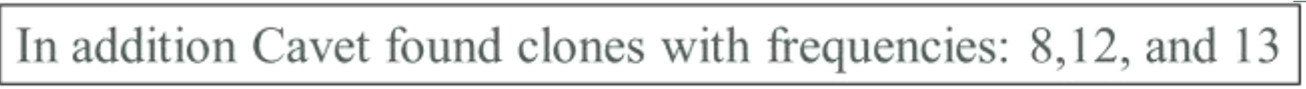
%LOC: Percent of lines with clones in them

%FILE: Percent of files with clones in them

Comparison of Clone Detectors

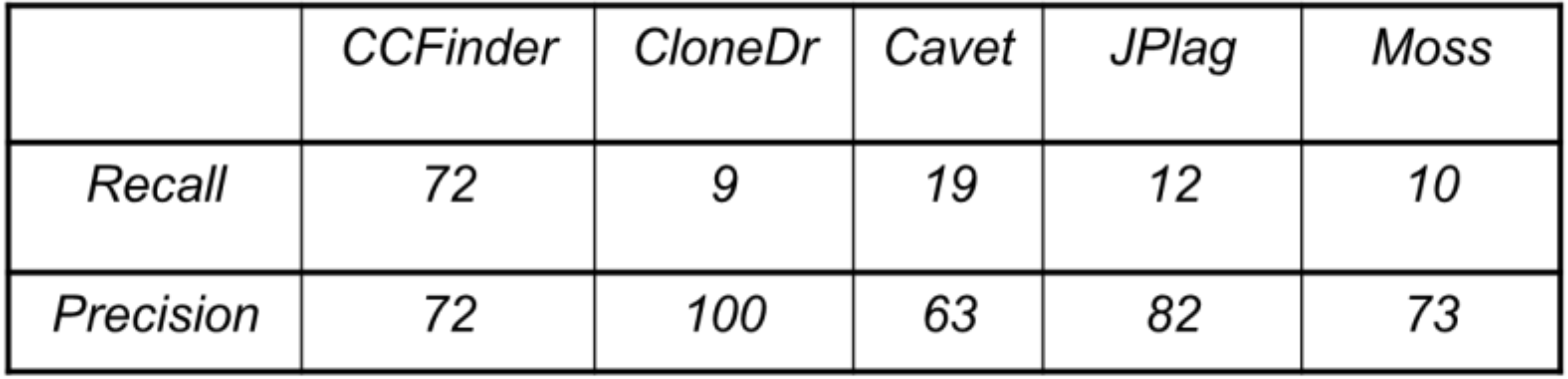


	CCFinder Token	CloneDr AST	Cavet Metric	Jplag Token	Moss Unknown
Frequency	CCFinder	CloneDr	Cavet	JPlag	Moss
1	569	66	40	95	104
2	98	6	34	10	8
3	33	2	13	4	0
4	14	0	6	1	0
5	16	0	5	0	0
6	19	0	5	0	0
7	2	0	1	0	0



In addition Cavet found clones with frequencies: 8,12, and 13

Comparison of Clone Detectors



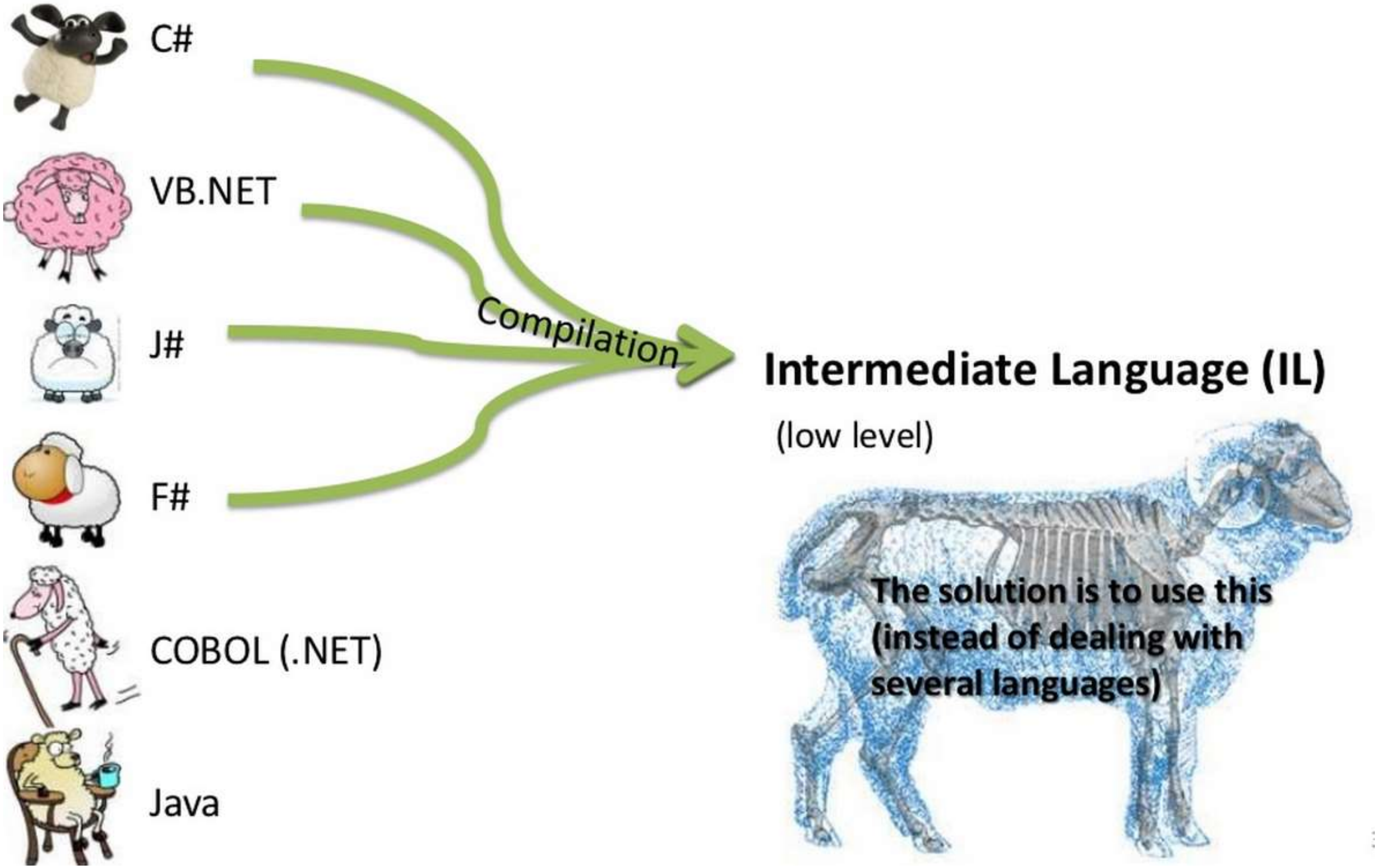
	CCFinder	CloneDr	Cavet	JPlag	Moss
Recall	*72*	*9*	*19*	*12*	*10*
Precision	*72*	*100*	*63*	*82*	*73*

Different code clone detectors find different clones

String based find direct clones
Token based find polymorphism issues and may be difficult to fix
Graph based find clones that can be automatically refactored

Clone Detection across Languages

General Solution



C#

VB.NET

J#

F#

COBOL (.NET)

Java

Compilation

Intermediate Language (IL)

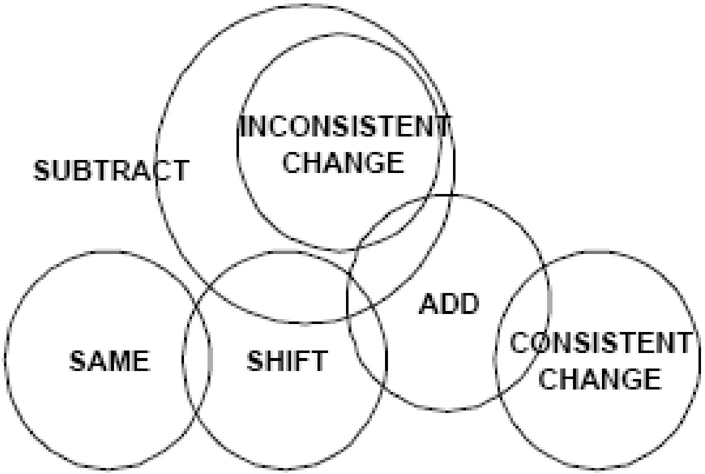
(low level)

The solution is to use this (instead of dealing with several languages)

Clone Management

Clone Genealogies

Refers to the life cycle of clones through various versions of a program.
Evolution Patterns of a code clone and their relationship are seen.



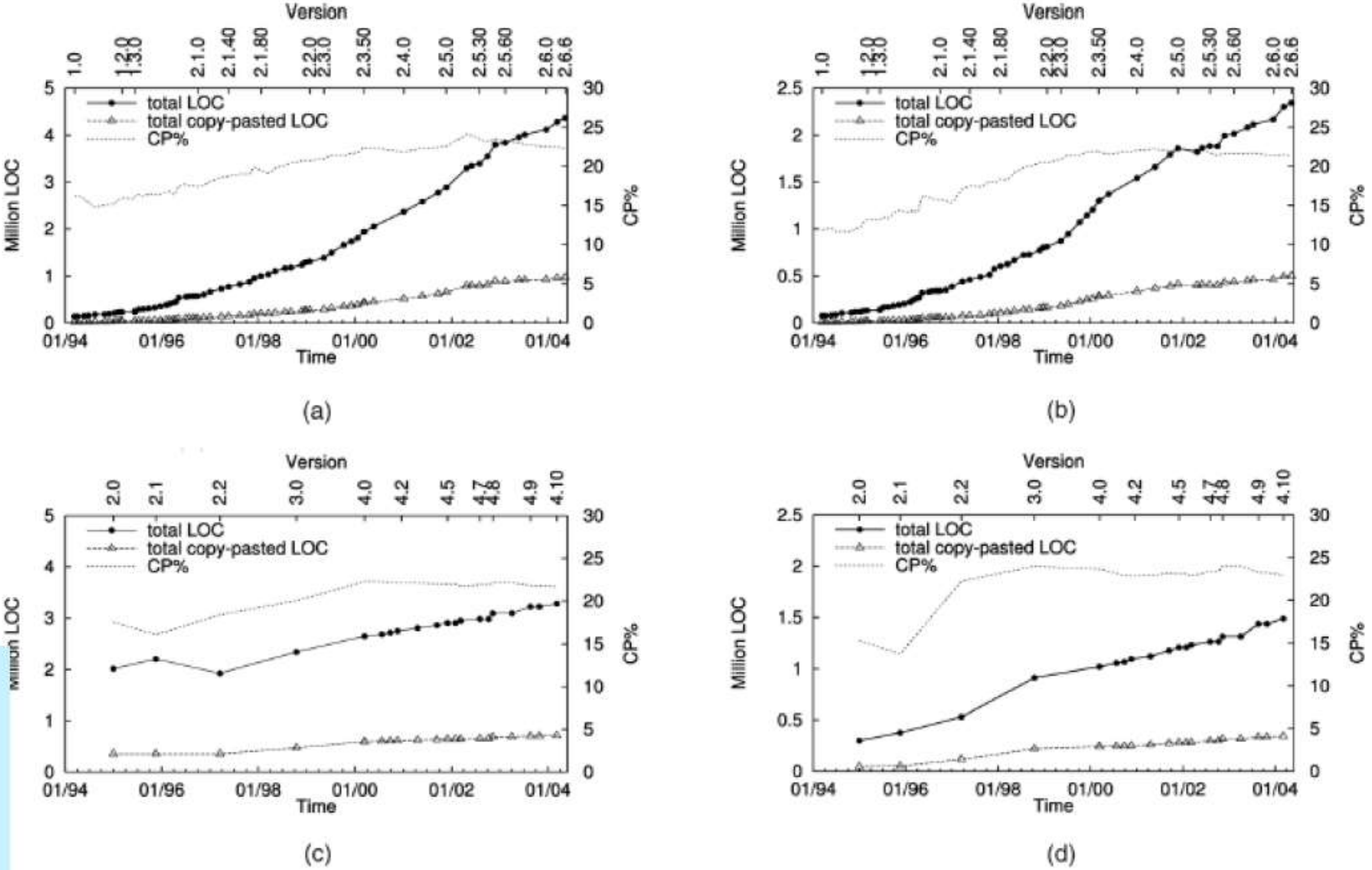
a) Linux

b) Linux "drivers"

c) Free BSD

d) Free BSD "sys"

Li et al. 2006



(a)

(b)

(c)

(d)

Increase followed by stabilization

Clone genealogy

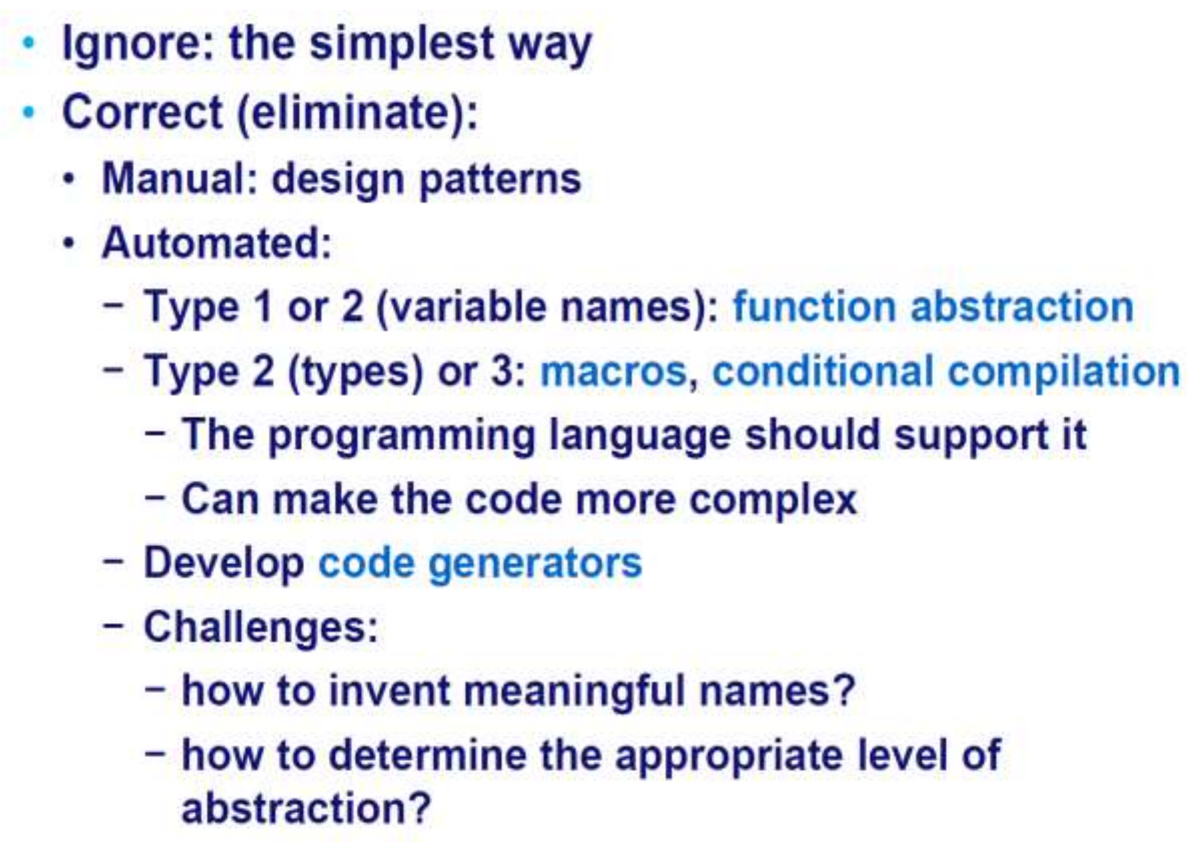
Consistently changing clones - all lineages in the clone genealogy include at least one "consistent change pattern"

Volatile clones – measured based on presence across various versions. "K-volatile"

Locally Non-refactorable clones – Programmer cannot refactor using pull-up or extract methods.

Long Lived Clones – Clones that lived across various versions of the program. Ideal for refactoring

What can we do about clones?



- Ignore: the simplest way
- Correct (eliminate):
  - Manual: design patterns
  - Automated:
    - Type 1 or 2 (variable names): function abstraction
    - Type 2 (types) or 3: macros, conditional compilation
      - The programming language should support it
      - Can make the code more complex
    - Develop code generators
    - Challenges:
      - how to invent meaningful names?
      - how to determine the appropriate level of abstraction?

Software Restructuring: General Idea

Developers continuously modify, enhance and adapt software.

As software evolves and strays away from its original design, three things happen.

- Decreased understandability
- Decreased reliability
- Increased maintenance cost

Decreased understandability is due to

- Increased complexity of code
- Out-of-date documentation
- Code not conforming to standards

SOFTWARE RESTRUCTURING: GENERAL IDEA

Decrease the complexity of software by improving its internal quality by restructuring the software.

Restructuring applied on object-oriented software is called refactoring.

Restructuring means reorganizing software (source code + documentation) to give it a different look, or structure.

Source code is restructured to improve some of its non-functional requirements:

- Readability
- Extensibility
- Maintainability
- Modularity

Restructuring **<u>does not modify</u>** the software’s functionalities.

Restructuring can be performed *while* adding new features.

SOFTWARE RESTRUCTURING: CORE IDEA

Software restructuring is informally stated as the modifications of software to make it

- easier to understand;
- easier to change;
- easier to change its documentation;
- less susceptible to faults when changes are made to it.

A higher level goal of restructuring is to increase the software value

- external software value: fewer faults in software is seen to be better by customers
- internal software value: a well-structured system is less expensive to maintain

Simple examples of restructuring

- Pretty printing
- Meaningful names for variables
- One statement per line of source code

SOFTWARE RESTRUCTURING: CORE IDEA

Developers and managers need to be aware of restructuring for the following reasons

- better understandability
- keep pace with new structures
- better reliability
- longer lifetime
- automated analysis

Characteristics of restructuring and refactoring

- The objective of restructuring and refactoring is to improve the internal and external values of software.
- Restructuring preserves the external behavior of the original program.
- Restructuring can be performed without adding new requirements.
- Restructuring generally produces a program in the same language.
  - Example: a C program is restructured into another C program.

Activities in a Refactoring Process

To restructure a software system, one follows a process with well defined activities.

- Identify what to refactor.
- Determine which refactorings to apply.
- Ensure that refactoring preserves the software's behavior.
- Apply the refactorings to the chosen entities.
- Evaluate the impacts of the refactorings.
- Maintain consistency.

The programmer identifies what to refactor from a set of high-level software artifacts.

- source code;
- design documents; and
- requirements documents.

Next, focus on specific portions of the chosen artifact for refactoring.

- Specific modules, functions, classes, methods, and data can be identified for refactoring.

Identify what to refactor

The concept of code smell is applied to source code to detect what should be refactored. (Fowler)

A code smell is any symptom in source code that possibly indicates a deeper problem.

Examples of code smell are:

- **duplicate code;**
- long parameter list;
- long methods;
- large classes;
- message chain.

Determine which refactorings to apply

Tool support is needed to identify a feasible subset of refactorings.

The following two techniques can be used to analyze a set of refactorings to select a feasible subset.

- Critical pair analysis
  - Given a set of refactorings, analyze each pair for conflicts. A apir is said to be conflicting if both of them cannot be applied together.
    - Example: R4 and R6 constitute a conflicting pair.
- Sequential dependency analysis
  - In order to apply a refactoring, one or more refactorings must be applied before.
  - If one refactoring has already been applied, a mutually exclusive refactoring cannot be applied anymore.
    - Example: after applying R1, R2, and R3, R4 becomes applicable. Now, if R4 is applied, then R6 is not applicable anymore.

Ensure that refactoring preserves the software’s behavior.

- Ideally, the input/output behavior of a program *after* refactoring is the same as the behavior *before* refactoring.
- In many applications, preservation of non-functional requirements is necessary.
- A non-exclusive list of such non-functional requirements is as follows:
  - Temporal constraints: A temporal constraint over a sequence of operations is that the operations occur in a certain order.
    - For real-time systems, refactorings should preserve temporal constraints.
  - Resource constraints: The software after refactoring does not demand more resources: memory, energy, communication bandwidth, and so on.
  - Safety constraints: It is important that the software does not lose its safety properties after refactoring.

Ensure that refactoring preserves the software’s behavior.

Two pragmatic ways of showing that refactoring preserves the software’s behavior.

Testing

- Exhaustively test the software *before* and *after* applying refactorings, and compare the observed behavior on a test-by-test basis.

Verification of preservation of call sequence

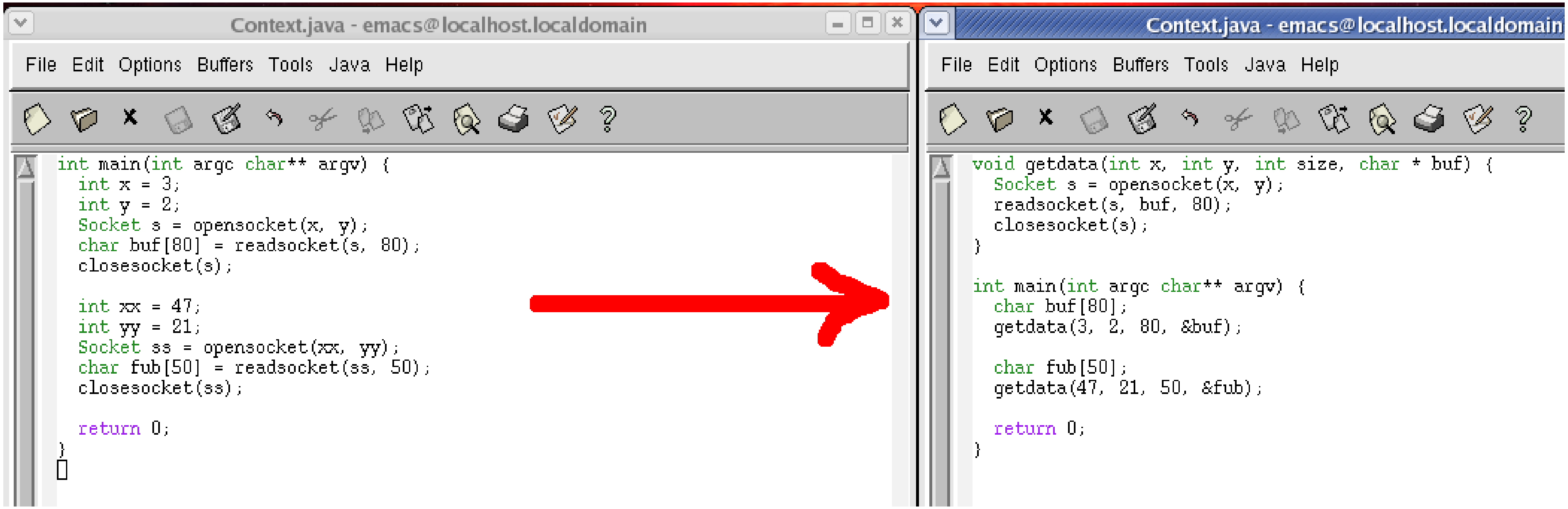
- Ensure that the sequence(s) of method calls are preserved in the refactored program.

Code Clone Refactoring

Use standard Refactoring methods

- “Extract” - Make a procedure
- “Pull Up” - Make an superclass

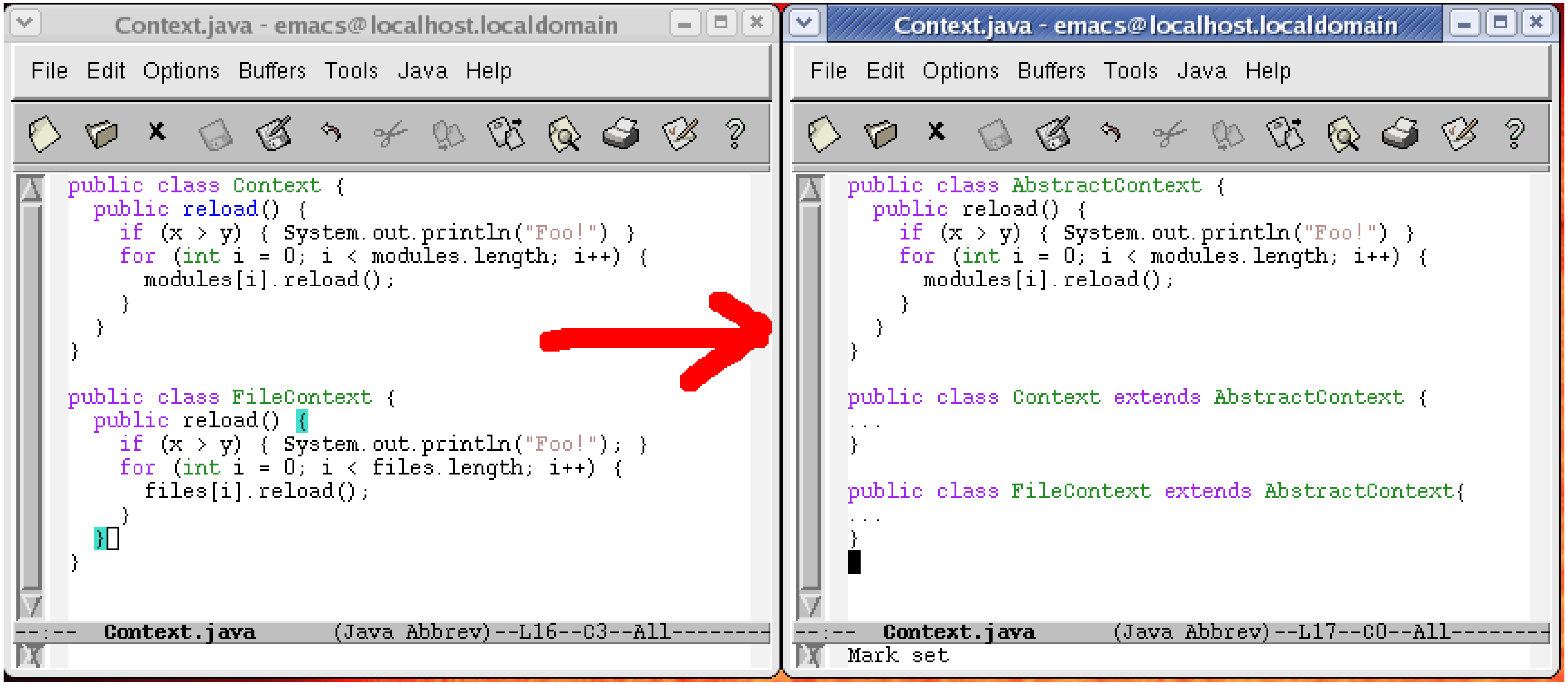
Refactoring of code clones advocated strongly by many practitioners including Fowler.


Context.java - emacs@localhost.localdomain
File Edit Options Buffers Tools Java Help
Context.java - emacs@localhost.localdomain
File Edit Options Buffers Tools Java Help

```
int main(int argc char** argv) {
  int x = 3;
  int y = 2;
  Socket s = opensocket(x, y);
  char buf[80] = readsocket(s, 80);
  closesocket(s);

  int xx = 47;
  int yy = 21;
  Socket ss = opensocket(xx, yy);
  char fub[50] = readsocket(ss, 50);
  closesocket(ss);

  return 0;
}
```

```
void getdata(int x, int y, int size, char * buf) {
  Socket s = opensocket(x, y);
  readsocket(s, buf, 80);
  closesocket(s);
}

int main(int argc char** argv) {
  char buf[80];
  getdata(3, 2, 80, &buf);

  char fub[50];
  getdata(47, 21, 50, &fub);

  return 0;
}
```

Extract


Context.java - emacs@localhost.localdomain
File Edit Options Buffers Tools Java Help
Context.java - emacs@localhost.localdomain
File Edit Options Buffers Tools Java Help

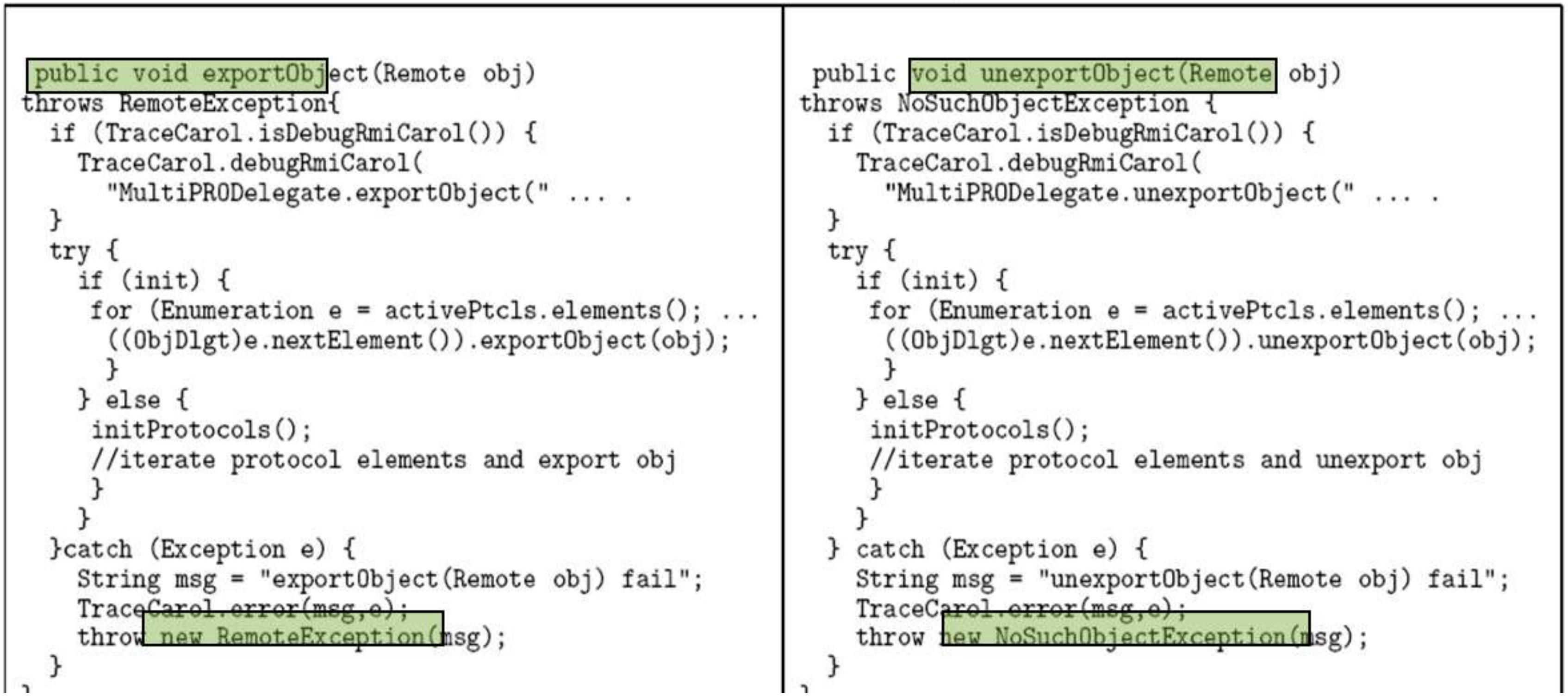
```
public class Context {
  public reload() {
    if (x > y) { System.out.println("Foo!") }
    for (int i = 0; i < modules.length; i++) {
      modules[i].reload();
    }
  }
}

public class FileContext {
  public reload() {
    if (x > y) { System.out.println("Foo!"); }
    for (int i = 0; i < files.length; i++) {
      files[i].reload();
    }
  }
}
```

```
--:-- Context.java        (Java Abbrev)--L16--C3--All--------
```

```
public class AbstractContext {
  public reload() {
    if (x > y) { System.out.println("Foo!") }
    for (int i = 0; i < modules.length; i++) {
      modules[i].reload();
    }
  }
}

public class Context extends AbstractContext {
...
}

public class FileContext extends AbstractContext{
...
}
```

```
--:-- Context.java        (Java Abbrev)--L17--C0--All--------
Mark set
```

Pull Up

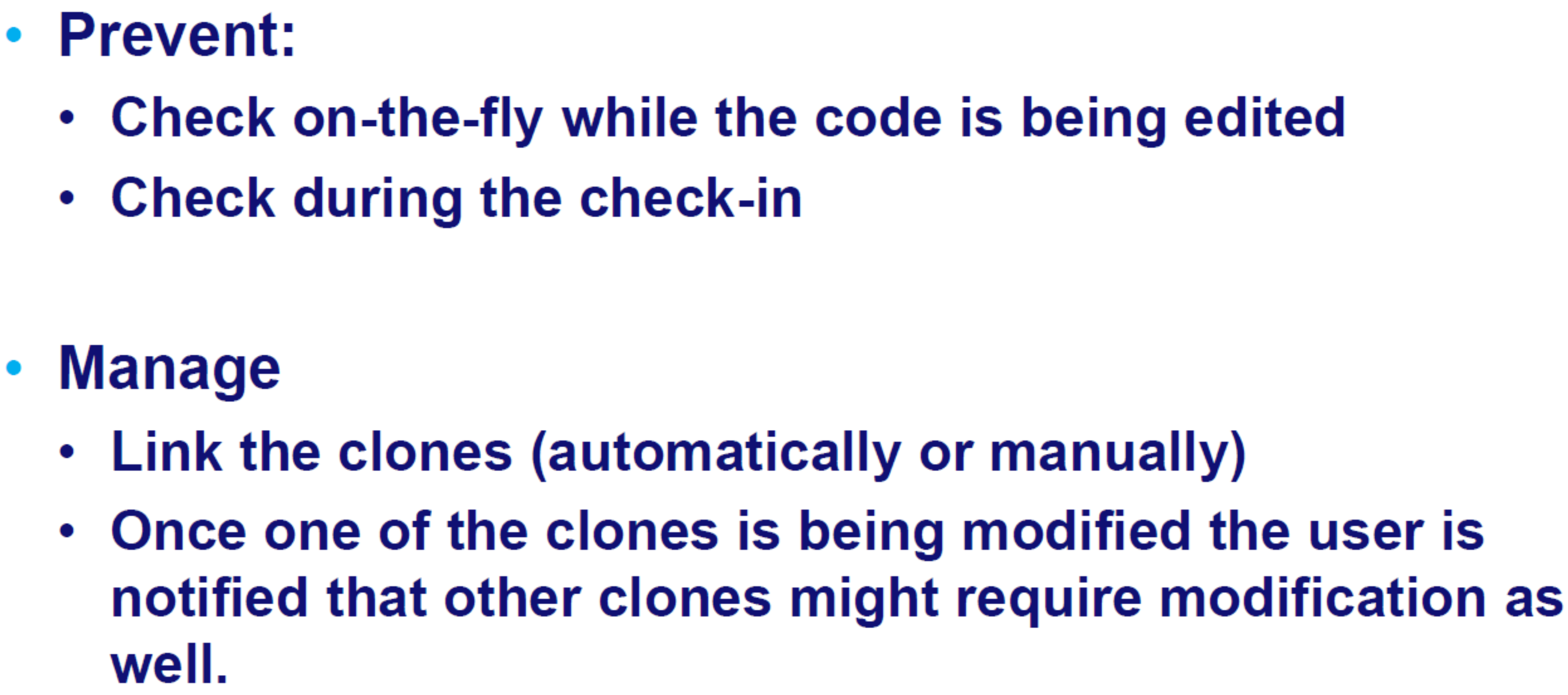


```
public void exportObject(Remote obj)
throws RemoteException{
  if (TraceCarol.isDebugRmiCarol()) {
    TraceCarol.debugRmiCarol(
      "MultiPRODelegate.exportObject(" ... .
  }
  try {
    if (init) {
     for (Enumeration e = activePtcls.elements(); ...
      ((ObjDlgt)e.nextElement()).exportObject(obj);
      }
    } else {
     initProtocols();
     //iterate protocol elements and export obj
     }
    }
  }catch (Exception e) {
    String msg = "exportObject(Remote obj) fail";
    TraceCarol.error(msg,e);
    throw new RemoteException(msg);
  }
}
```

```
 public void unexportObject(Remote obj)
throws NoSuchObjectException {
  if (TraceCarol.isDebugRmiCarol()) {
    TraceCarol.debugRmiCarol(
      "MultiPRODelegate.unexportObject(" ... .
  }
  try {
    if (init) {
     for (Enumeration e = activePtcls.elements(); ...
      ((ObjDlgt)e.nextElement()).unexportObject(obj);
      }
    } else {
     initProtocols();
     //iterate protocol elements and unexport obj
     }
    }
  } catch (Exception e) {
    String msg = "unexportObject(Remote obj) fail";
    TraceCarol.error(msg,e);
    throw new NoSuchObjectException(msg);
  }
}
```

Unfactorable Code Clone

What Else can we do about Clones (cont.)



- **Prevent:**
  - **Check on-the-fly while the code is being edited**
  - **Check during the check-in**

- **Manage**
  - **Link the clones (automatically or manually)**
  - **Once one of the clones is being modified the user is notified that other clones might require modification as well.**

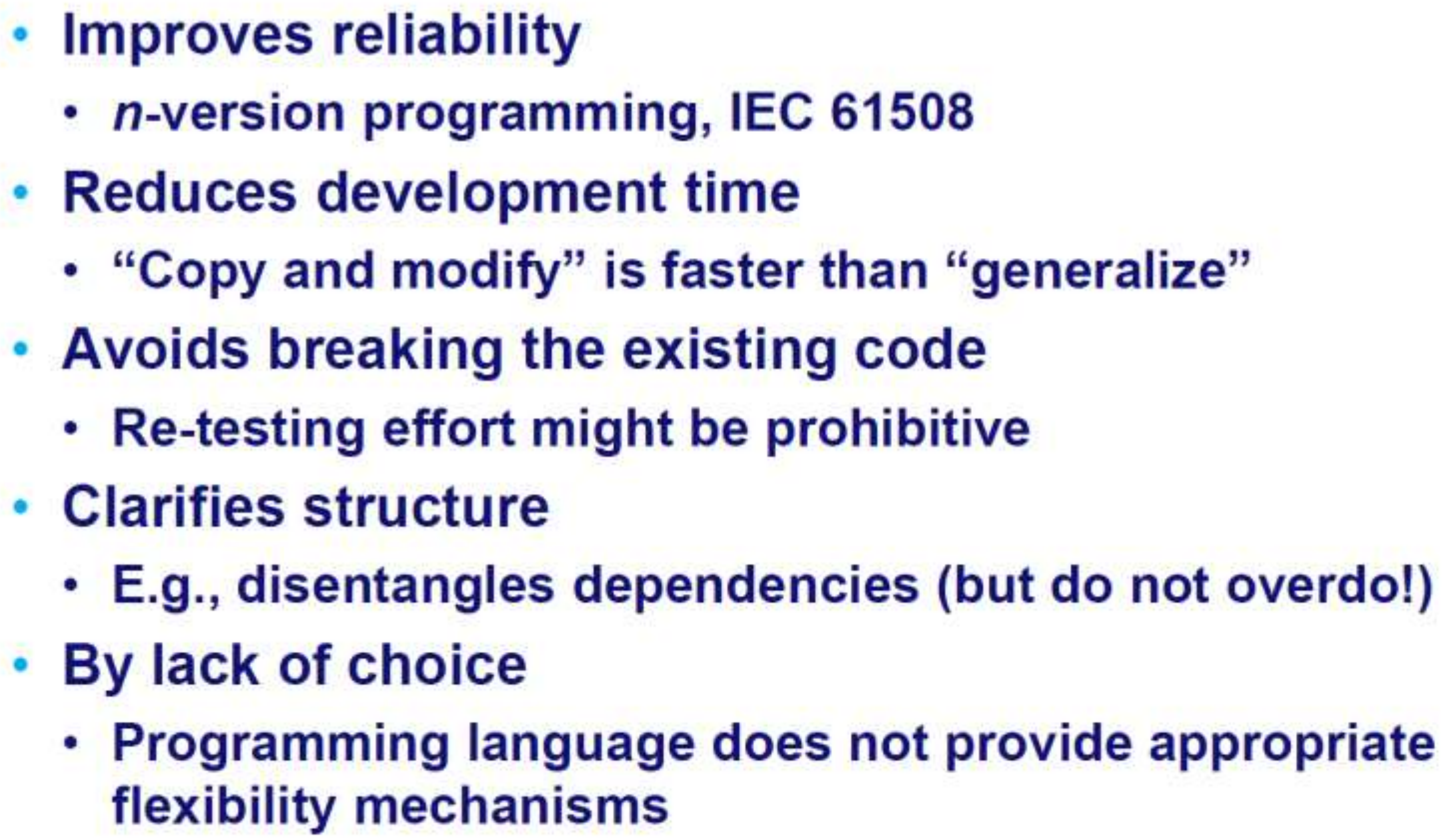
Controversial statements

Clones should **be avoided**, no matter what !!

Clones **are good**! We should not remove all clones !!

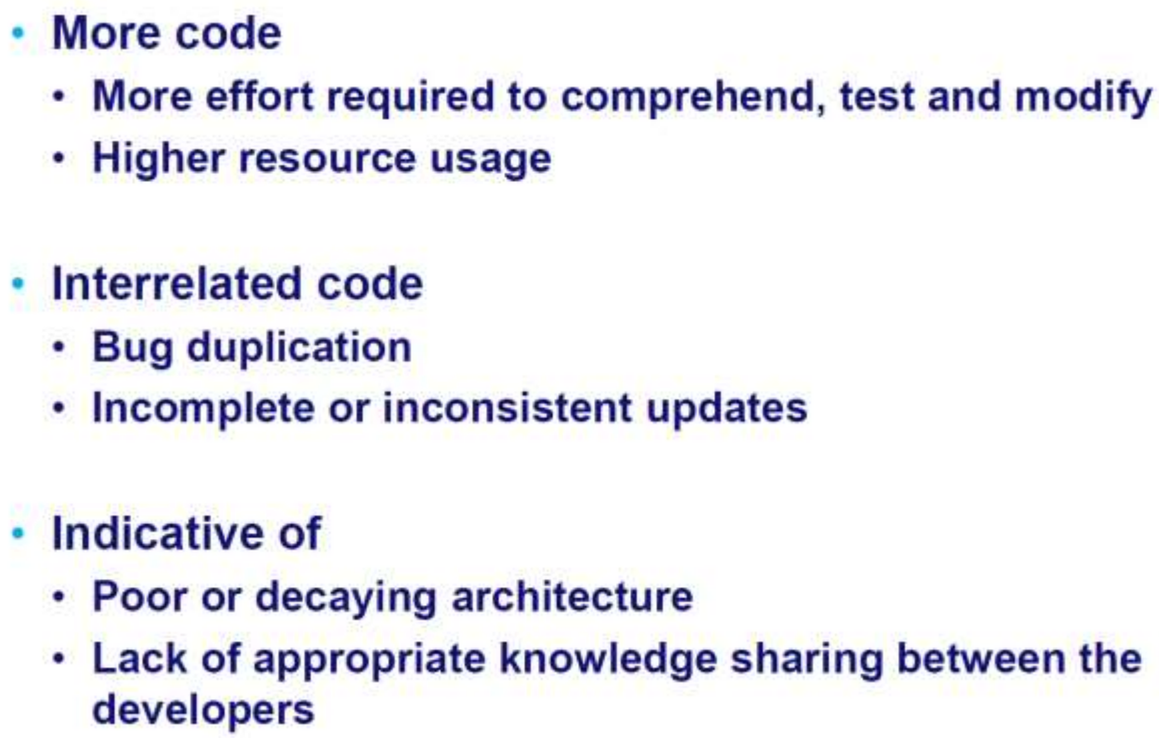
Clones/code quality/maintenance is there any relationship?

Why Clones can be good



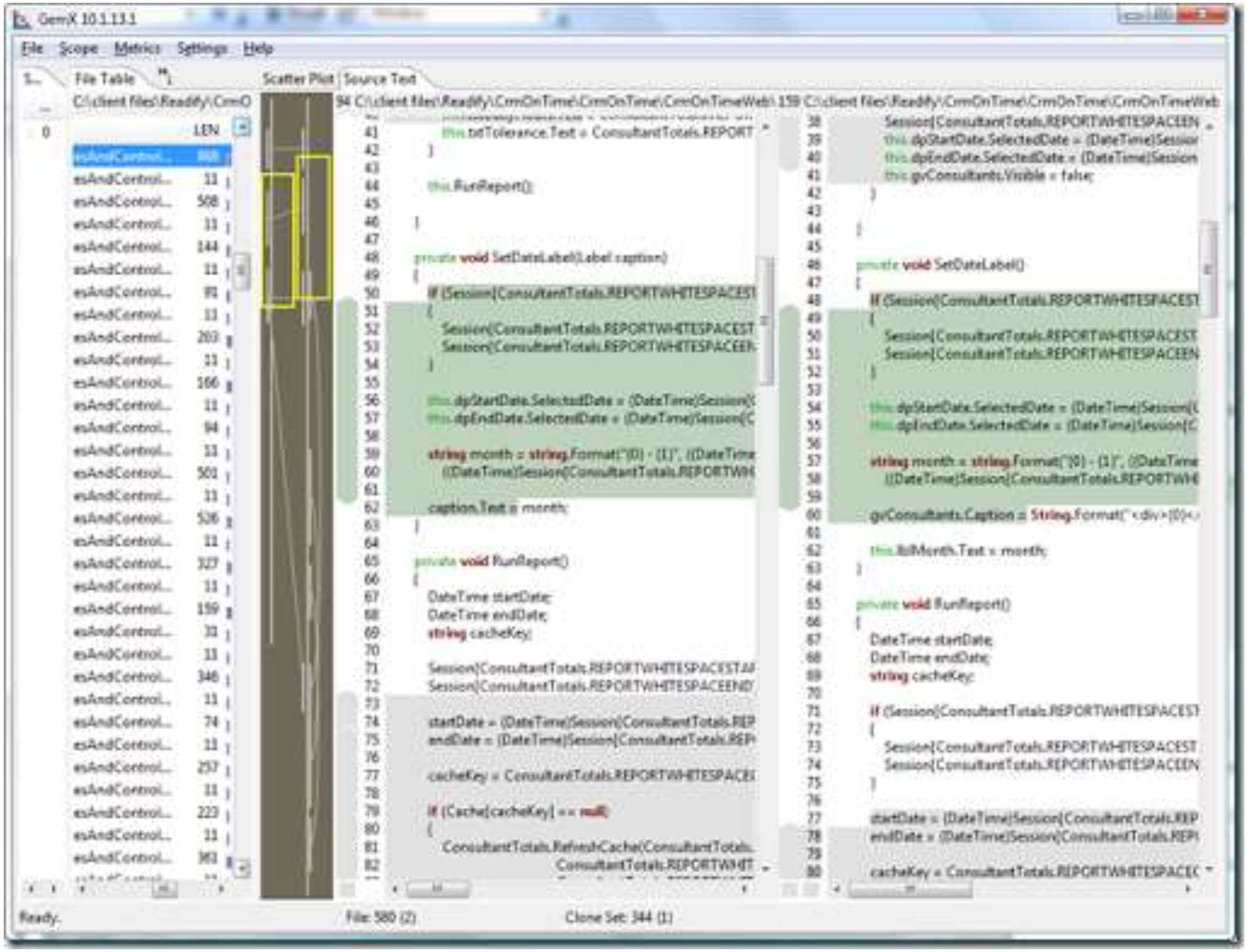
- Improves reliability
  - *n*-version programming, IEC 61508
- Reduces development time
  - “Copy and modify” is faster than “generalize”
- Avoids breaking the existing code
  - Re-testing effort might be prohibitive
- Clarifies structure
  - E.g., disentangles dependencies (but do not overdo!)
- By lack of choice
  - Programming language does not provide appropriate flexibility mechanisms

Bad News



- More code
  - More effort required to comprehend, test and modify
  - Higher resource usage
- Interrelated code
  - Bug duplication
  - Incomplete or inconsistent updates
- Indicative of
  - Poor or decaying architecture
  - Lack of appropriate knowledge sharing between the developers

CLONE DETECTION TOOLS



C:\client files\Readify\CrmO

	LEN
esAndControl...	868
esAndControl...	11
esAndControl...	508
esAndControl...	11
esAndControl...	144
esAndControl...	11
esAndControl...	91
esAndControl...	11
esAndControl...	203
esAndControl...	11
esAndControl...	166
esAndControl...	11
esAndControl...	94
esAndControl...	11
esAndControl...	501
esAndControl...	11
esAndControl...	526
esAndControl...	11
esAndControl...	327
esAndControl...	11
esAndControl...	159
esAndControl...	31
esAndControl...	11
esAndControl...	346
esAndControl...	11
esAndControl...	74
esAndControl...	11
esAndControl...	257
esAndControl...	11
esAndControl...	223
esAndControl...	11
esAndControl...	363

94 C:\client files\Readify\CrmOnTime\CrmOnTime\CrmOnTimeWeb\

```
            this.txtTolerance.Text = ConsultantTotals.REPORT
        }

        this.RunReport();

    }

    private void SetDateLabel(Label caption)
    {
        if (Session[ConsultantTotals.REPORTWHITESPACEST
        {
            Session[ConsultantTotals.REPORTWHITESPACEST
            Session[ConsultantTotals.REPORTWHITESPACEEN
        }

        this.dpStartDate.SelectedDate = (DateTime)Session[C
        this.dpEndDate.SelectedDate = (DateTime)Session[C

        string month = string.Format("{0} - {1}", ((DateTime
            ((DateTime)Session[ConsultantTotals.REPORTWH

        caption.Text = month;
    }

    private void RunReport()
    {
        DateTime startDate;
        DateTime endDate;
        string cacheKey;

        Session[ConsultantTotals.REPORTWHITESPACESTAR
        Session[ConsultantTotals.REPORTWHITESPACEEND

        startDate = (DateTime)Session[ConsultantTotals.REP
        endDate = (DateTime)Session[ConsultantTotals.REP

        cacheKey = ConsultantTotals.REPORTWHITESPACE

        if (Cache[cacheKey] == null)
        {
            ConsultantTotals.RefreshCache(ConsultantTotals.
                                ConsultantTotals.REPORTWHIT
```

159 C:\client files\Readify\CrmOnTime\CrmOnTime\CrmOnTimeWeb

```
            Session[ConsultantTotals.REPORTWHITESPACEEN
            this.dpStartDate.SelectedDate = (DateTime)Session
            this.dpEndDate.SelectedDate = (DateTime)Session
            this.gvConsultants.Visible = false;
        }

    }

    private void SetDateLabel()
    {
        if (Session[ConsultantTotals.REPORTWHITESPACEST
        {
            Session[ConsultantTotals.REPORTWHITESPACEST
            Session[ConsultantTotals.REPORTWHITESPACEEN
        }

        this.dpStartDate.SelectedDate = (DateTime)Session[C
        this.dpEndDate.SelectedDate = (DateTime)Session[C

        string month = string.Format("{0} - {1}", ((DateTime
            ((DateTime)Session[ConsultantTotals.REPORTWH

        gvConsultants.Caption = String.Format("<div>{0}</

        this.lblMonth.Text = month;
    }

    private void RunReport()
    {
        DateTime startDate;
        DateTime endDate;
        string cacheKey;

        if (Session[ConsultantTotals.REPORTWHITESPACEST
        {
            Session[ConsultantTotals.REPORTWHITESPACEST
            Session[ConsultantTotals.REPORTWHITESPACEEN
        }

        startDate = (DateTime)Session[ConsultantTotals.REP
        endDate = (DateTime)Session[ConsultantTotals.REP

        cacheKey = ConsultantTotals.REPORTWHITESPACE(
```

Ready.

File: 580 (2)

Clone Set: 344 (1)

CCFinder

Simian

Found 6 duplicate lines in the following files:

Between lines 201 and 207 in simian/build/dist/src/java/awt/image/WritableRaster.java

Between lines 1305 and 1311 in simian/build/dist/src/java/awt/image/Raster.java

Found 6 duplicate lines in the following files:

Between lines 920 and 926 in simian/build/dist/src/com/sun/imageio/plugins/jpeg/JFIFMarkerSegment.java

Between lines 908 and 914 in simian/build/dist/src/com/sun/imageio/plugins/jpeg/JFIFMarkerSegment.java

…