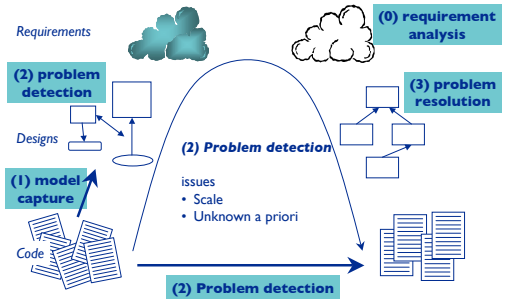
6. Code Duplication

a.k.a. Software Cloning, Copy&Paste Programming

- ***Code Duplication***
  - + What is it?
  - + Why is it harmful?
- Detecting Code Duplication
- Approaches
- A Lightweight Approach
- Visualization (dotplots)
- Duplo

The Reengineering Life-Cycle



Code is Copied

Small Example from the Mozilla Distribution (Milestone 9)
Extract from /dom/src/base/nsLocation.cpp

```
[432]  NS_IMETHODIMP                                    [467]  NS_IMETHODIMP                                    [497]  NS_IMETHODIMP
[433]  LocationImpl::GetPathname(nsString& aPathname)   [468]  LocationImpl::SetPathname(const nsString&      [498]  LocationImpl::GetPort(nsString& aPort)
[434]  {                                                [469]  {                                                [499]  {
[435]    nsAutoString href;                             [470]    nsAutoString href;                             [500]    nsAutoString href;
[436]    nsIURI *url;                                   [471]    nsIURI *url;                                   [501]    nsIURI *url;
[437]    nsresult result = NS_OK;                       [472]    nsresult result = NS_OK;                       [502]    nsresult result = NS_OK;
[438]                                                   [473]                                                   [503]
[439]    result = GetHref(href);                        [474]    result = GetHref(href);                        [504]    result = GetHref(href);
[440]    if (NS_OK == result) {                         [475]    if (NS_OK == result) {                         [505]    if (NS_OK == result) {
[441]  #ifndef NECKO                                    [476]  #ifndef NECKO                                    [506]  #ifndef NECKO
[442]    result = NS_NewURI(&url, href);                [477]    result = NS_NewURI(&url, href);                [507]    result = NS_NewURI(&url, href);
[443]  #else                                            [478]  #else                                            [508]  #else
[444]    result = NS_NewURI(&url, href);                [479]    result = NS_NewURI(&url, href);                [509]    result = NS_NewURI(&url, href);
[445]  #endif // NECKO                                  [480]  #endif // NECKO                                  [510]  #endif // NECKO
[446]    if (NS_OK == result) {                         [481]    if (NS_OK == result) {                         [511]    if (NS_OK == result) {
[447]  #ifdef NECKO                                     [482]      char *buf = aPathname.ToNewCString();        [512]      aPort.SetLength(0);
[448]      char* file;                                  [483]  #ifdef NECKO                                     [513]  #ifdef NECKO
[449]      result = url->GetPath(&file);                [484]      url->SetPath(buf);                           [514]      PRInt32 port;
[450]  #else                                            [485]  #else                                            [515]      (void)url->GetPort(&port);
[451]      const char* file;                            [486]      url->SetFile(buf);                           [516]  #else
[452]      result = url->GetFile(&file);                [487]  #endif                                           [517]      PRUint32 port;
[453]  #endif                                           [488]      SetURL(url);                                 [518]      (void)url->GetHostPort(&port);
[454]      if (result == NS_OK) {                       [489]      delete[] buf;                                [519]  #endif
[455]        aPathname.SetString(file);                 [490]      NS_RELEASE(url);                             [520]      if (-1 != port) {
[456]  #ifdef NECKO                                     [491]    }                                              [521]        aPort.Append(port, 10);
[457]        nsCRT::free(file);                         [492]  }                                                [522]      }
[458]  #endif                                           [493]                                                   [523]      NS_RELEASE(url);
[459]      NS_IF_RELEASE(url);                          [494]  return result;                                   [524]    }
[460]    }                                              [495]  }                                                [525]  }
[461]  }                                                [496]                                                   [526]
[462]                                                                                                           [527]  return result;
[463]  return result;                                                                                          [528]  }
[464]  }                                                                                                       [529]
[465]
[466]
```


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 Considered To Be Copied 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 artefacts range from expressions, to functions, to data structures, and to entire subsystems.

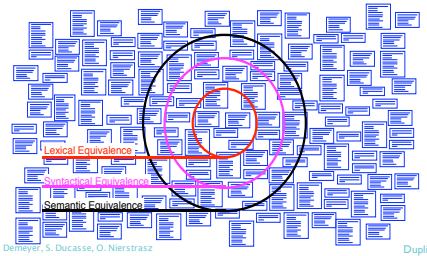
Copied Code Problems

- 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
- Metaphorically speaking:
  - + Software Aging, "hardening of the arteries",
  - + "Software Entropy" increases even small design changes become very difficult to effect

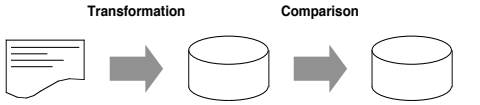
Code Duplication Detection

Nontrivial problem:

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



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

A Lightweight Approach

- Code Duplication
  - + What is it?
  - + Why is it harmful?
- Detecting Code Duplication
- Approaches
- ***A Lightweight Approach***
- Visualization (dotplots)
- Duplo

Simple Detection Approach (i)

- **Assumption**:
  - Code segments are just copied and changed at a few places
- **Code Transformation Step**
  - remove white space, comments
  - remove lines that contain uninteresting code elements (e.g., just 'else' or '}')

```
...
//assign same fastid as container
fastid = NULL;
const char* fidptr = get_fastid();
if(fidptr != NULL) {
  int l = strlen(fidptr);
  fastid = newchar[ l + 1 ];
```

```
...
fastid=NULL;
constchar*fidptr=get_fastid();
if(fidptr!=NULL)
intl=strlen(fidptr)
fastid = newchar[l+]
```


Simple Detection Approach (ii)

- **Code Comparison Step**
  - + Line based comparison (Assumption: Layout did not change during copying)
  - + Compare each line with each other line.
  - + Reduce search space by hashing:
    - *1. Preprocessing: Compute the hash value for each line*
    - *2. Actual Comparison: Compare all lines in the same hash bucket*
- **Evaluation of the Approach**
  - + Advantages: Simple, language independent
  - + Disadvantages: Difficult interpretation

A Perl script for C++ (i)

```
$equivalenceClassMinimalSize = 1;
$slidingWindowSize          = 5;
$removeKeywords             = 0;
@keywords      = qw(if
   then
   else
  );

$keywordsRegExp = join '|', @keywords;

@unwantedLines = qw( else
    return
    return;
    {
    }
    ;
 );
push @unwantedLines, @keywords;
```

```
while (<>) {
  chomp;
  $totalLines++;

  # remove comments of type /* */
  my $codeOnly = '';
  while(($inComment  && m|\*/|) ||
(!$inComment && m|/\*|)) {
    unless($inComment) { $codeOnly .= $` }
    $inComment = !$inComment;
    $_ = $';
  }
  $codeOnly .= $_  unless $inComment;
  $_ = $codeOnly;

  s|//.*$||; # remove comments of type //
  s/\s+//g; #remove white space
  s/$keywordsRegExp//og if
$removeKeywords;  #remove keywords
```


A Perl script for C++ (ii)

```
$codeLines++;
  push @currentLines, $_;
  push @currentLineNos, $.;
  if($slidingWindowSize < @currentLines) {
    shift @currentLines;
    shift @currentLineNos;}
  #print STDERR "Line $totalLines >$_<\n";
  my $lineToBeCompared      = join '',  @currentLines;
  my $lineNumbersCompared   = "<$ARGV>"; # append
the name of the file
   $lineNumbersCompared   .= join '/', @currentLineNos;
   #print STDERR "$lineNumbersCompared\n";
  if($bucketRef = $eqLines{$lineToBeCompared}) {
    push @$bucketRef,  $lineNumbersCompared;
  } else {$eqLines{$lineToBeCompared} = [
$lineNumbersCompared ];}
if(eof) { close ARGV } # Reset linenumber-count for next
file
```

- Handles multiple files
- Removes comments and white spaces
- Controls noise (if, {,)
- Granularity (number of lines)
- Possible to remove keywords

Output Sample

```
Lines:
create_property(pd,pnImplObjects,stReference,false,*iImplObjects);
create_property(pd,pnElttype,stReference,true,*iEltType);
create_property(pd,pnMinelt,stInteger,true,*iMinelt);
create_property(pd,pnMaxelt,stInteger,true,*iMaxelt);
create_property(pd,pnOwnership,stBool,true,*iOwnership);
Locations:
</face/typesystem/SCTypesystem.C>6178/6179/6180/6181/6182
</face/typesystem/SCTypesystem.C>6198/6199/6200/6201/6202
Lines:
create_property(pd,pnSupertype,stReference,true,*iSupertype);
create_property(pd,pnImplObjects,stReference,false,*iImplObjects);
create_property(pd,pnElttype,stReference,true,*iEltType);
create_property(pd,pMinelt,stInteger,true,*iMinelt);
create_property(pd,pnMaxelt,stInteger,true,*iMaxelt);
Locations: </face/typesystem/SCTypesystem.C>6177/6178
</face/typesystem/SCTypesystem.C>6229/6230
```

Lines = duplicated lines

Locations = file names and line number

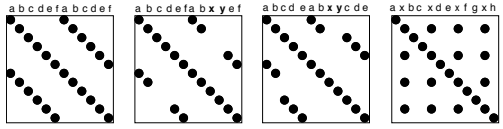
Enhanced Simple Detection Approach

- Code Comparison Step
  - + Same as before +
    - Collect consecutive matching lines into match sequences
    - Allow holes in the match sequence
- Evaluation of the Approach
  - + Advantages
    - Identifies more real duplication, language independent
  - + Disadvantages
    - Less simple
    - Misses copies with (small) changes on every line

Visualization of Duplicated Code

- Visualization provides insights into the duplication situation
- A simple version can be implemented in three days
- Scalability issue

- Dotplots — Technique from DNA Analysis
  - Code is put on vertical as well as horizontal axis
  - A match between two elements is a dot in the matrix

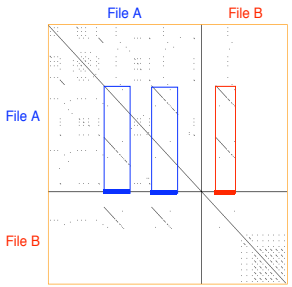


Visualization of Copied Code Sequences

Detected Problem
File A contains two copies of a piece of code

File B contains another copy of this code

Possible Solution
Extract Method

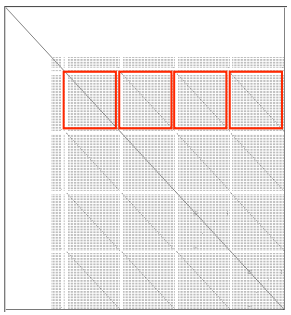


All examples are made using Duploc from an industrial case study (1 Mio LOC C++ System)

Visualization of Repetitive Structures

Detected Problem
4 Object factory clones: a switch statement over a type variable is used to call individual construction code

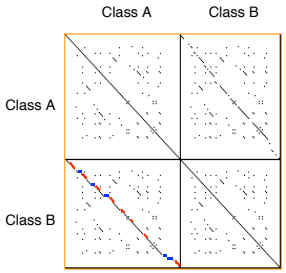
Possible Solution
Strategy Method



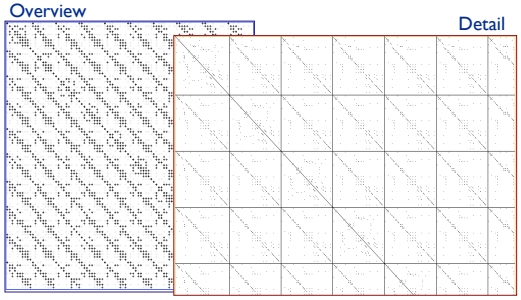
Visualization of Cloned Classes

Detected Problem:
Class A is an edited copy of class B. Editing & Insertion

Possible Solution
Subclassing …



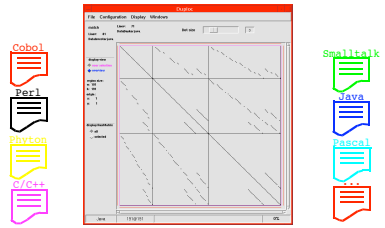
Visualization of Clone Families



20 Classes implementing lists for different data types

Lightweight is sometimes not enough

Duploc is scalable, integrates detection and visualization



It runs really everywhere (Smalltalk inside)
⇒ *Contact us if you need it!*

Conclusion

- Duplicated code is a real problem
  - + makes a system progressively harder to change
- Detecting duplicated code is a hard problem
  - + some simple technique can help
  - + tool support is needed
- Visualization of code duplication is useful
  - + some basic support are easy to build
  - + one student build a simple visualization tool in three days
- Curing duplicated code is an active research area