Reengineering Analysis of Object-Oriented Systems via Duplication Analysis

F. Fioravanti, G. Migliarese, P. Nesi

Department of Systems and Informatics
University of Florence
Via S. Marta 3, 50139, Firenze, Italy
tel: +39-055-4796523, fax: +39-055-4796363
nesi@ingfi1.ing.unifi.it, nesi@dsi.unifi.it
http://www.dsi.unifi.it/~nesi
Problem Description

- **Different OO software applications** evolved on the basis of the same initial version of an OO Class Library or FW

- **Re-engineering the**
  - library (generalising, improving, etc.) and
  - applications (sharing more code with the library)

- **Via Code Analysis looking for duplications**
  - manual/assisted inspection for duplication and reasoning
  - duplication analysis with TREND tool
  - comparison of these processes
Problem Description, Why and Effects

- **Reengineering performed to**
  - extend the library/framework application domain
  - reduce costs of maintenance, of applications and of library
  - increase maintainability, ...
  - reduce the fault proneness, ....

- **Unfortunately, high costs for code reengineering**
  - understanding
  - navigation
  - processing:
    - method generalisation, moving attributes, changing class hierarchy, etc.
Duplications and Similarities in OO systems

- **Functional Duplications (cut and past !!)**
  - Self duplication of code segments (methods and/or classes)
  - duplication of methods instead of parameterisation
  - similar methods in different classes, different types

- **Structural Similarities**
  - similar classes in terms of data structure
  - classes with different names, modelling “similar” objects
  - class attributes of the same type with different names for “modelling the same aspects”
Duplication Analysis and Metrics for OO

- Duplication analysis can be influenced by
  - #… solving
  - format dependency
    - blank lines, comments, etc.
  - single line vs small sequences of duplicated lines
  - variable and their types
  - computationally intensive
    - polishing code, standardizing formatting

- To reduce these problems: pre-processing and specific metrics and a method for OO systems
  - detecting duplications at levels of
    - files, classes and methods
  - class similarity on class structure and hierarchy
Research Tool: TRENDEn

- **support for the Reengineering Process**
  - Code Analysis for object oriented code

- **Several different algorithms and metrics for**
  - pre-processing
  - duplication analysis
  - similarity detection

- **Reasoning on results, Method proposed**
  - support of visualisation methods

- **Reorganisation of classes/methods in modules**
  - classes and methods assignment to modules, packages

- **code indexing for**
  - fast manual precise reengineering
General TREND Process Architecture

Preprocessing:
- # elimination
- Reformatting
- metric estimation
- code reorganization
- variable -> type

Preprocessing:
- General information on data variables and constants
- Structural Metrics on data: attributes, classes, structures, unions, templates.
- Metrics code flow (methods and procedure)

Sources

General TREND Process Architecture
Pre-processing algorithms

- **SPA, Source Processing Algorithm**
  - Comment elimination, preprocessing phase of compiler

- **SFA, Source Formatting Algorithm**
  - one instruction per line ({ and } on different lines)
  - *structural metrics estimation*
  - *code reorganization, one Hxx + one Cxx per class*
  - *flow charts of methods and procedures*

- **VSA, Variable Substitution Algorithm**
  - substitution of Variable names with their type/class name
Code analysis, Duplication analysis

- **Algorithms for Duplication Estimation**
  - line duplication at level of files, class and methods
  - sequence of duplicated lines (minimum number of duplicated lines to consider a duplication)
  - estimation of duplication metrics

- **Extraction of structural and functional aspects**
  - structural similarities of classes
  - metric values about similarities

- **Results analysis**
  - some guidelines for reasoning on tables
  - visualisation mechanisms
  - code indexing for code navigation
General and Structural Metrics

- **TNAL** Total Number of Locally defined Attributes
- **TLOC** Total number of Lines Of Code
- **NCL** Number of system classes
- **TNML** Total Number of Locally defined Methods
- **Nbyte** Total Number of bytes of the system source files
- **NFile** Total Number of system source Files

- **NAL\_i** Number of local attributes of the i-th class
- **NALST\_ij** (Number of Attributes Locally defined with the Same Type)
  number of identical in type attributes between two classes, independently of the access qualifier (i.e., private, protected and public)
Metrics for Duplication Analysis

- \( NL_i \) Number of Lines of entity \( X_i \)
- \( NLID_{ij} \) (Number of Lines Duplicated) number of code lines of \( X_i \) which are also present in \( X_j \)
- \( NLIDS_{ij} \) (Number of Lines In Duplicated Sequences) length of the sequence of lines of \( X_i \) that are also present in \( X_j \)

*In the next tables the number of consecutive lines was set to 3*

- \( IID_{ij} = 100 \frac{NLID_{ij}}{NL_i} \) (Identity Index of Duplication) Duplication index of \( X_i \) with respect to \( X_j \)
- \( IID_{Sij} = 100 \frac{NLIDS_{ij}}{NL_i} \) (Identity Index of Duplication in Sequences) Duplication index of \( X_i \) with respect to \( X_j \), by considering only the sequences of duplicated lines
- \( IID_{ij} \) typically different than \( IID_{ji} \), etc.

- \( HM = \frac{1}{\text{mean}\left\{\frac{1}{a_1},...,\frac{1}{a_n}\right\}} \) **HM** Harmonic Mean of the above \( IID, IID_S \) metrics

- \( \forall i,j \in S_1 \cup S_2: \quad SI = \text{mean}\{IID_{ij}\} \) System Identity on NxM, (N+M)x(N+M) values
Reengineering Process: Case Study

- **two applications:** MWB and VM
  - based on the same class library
  - developed by different teams
  - based on the same application domain

- **Reengineering to obtain**
  **a New Library and New Applications**
  - increment of maintainability, less code, larger library
  - investment for new applications

- **Manual reengineering**
  - Adoption of a simple tool for duplication detection, no pre-processing and no metrics.
  - skilled people

- Results compared with those obtained by using the proposed metrics and tool, TREND
Performed Analyses

Duplication estimation may include the estimation of self duplications.

Rewritten: Duplication estimation may include the estimation of self duplications.

Reengineering analysis to assess the work performed by the team.

Rewritten: Reengineering analysis to assess the work performed by the team.

Domain Analysis
Domain or framework/library

Evolution

System A
System B

Reengineering

New System A
New System B

Reengineering Process
Manual Re-engineering Results

- 20 classes have been changed and some lines added

<table>
<thead>
<tr>
<th>System</th>
<th>NFile</th>
<th>NCL</th>
<th>TNML</th>
<th>TNAL</th>
<th>TLOC</th>
<th>NBYTE</th>
</tr>
</thead>
<tbody>
<tr>
<td>MWB/VM old</td>
<td>7</td>
<td>76</td>
<td>589</td>
<td>674</td>
<td>12.016</td>
<td>389.921</td>
</tr>
<tr>
<td>MWB/VM new</td>
<td>10</td>
<td>62</td>
<td>530</td>
<td>640</td>
<td>11.708</td>
<td>415.659</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>OPERATION PERFORMED</th>
<th>NUMBER OF CLASSES</th>
</tr>
</thead>
<tbody>
<tr>
<td>Deleted</td>
<td>14</td>
</tr>
<tr>
<td>Modified and moved in the library</td>
<td>13</td>
</tr>
<tr>
<td>Modified</td>
<td>7</td>
</tr>
<tr>
<td>Moved in the library</td>
<td>10</td>
</tr>
<tr>
<td>Unchanged</td>
<td>32</td>
</tr>
</tbody>
</table>

MWB old

- MO_MWB.hxx
- MO_MWB.cxx
- MO_list.hxx
- MN_MWB.hxx
- MN_MWB.cxx

VM old

- MO_VM.hxx
- MO_VM.cxx
- Other files

MWB new

VM new
Problems Remained from Manual Reengineering.

- Some duplications were still present in the new applications:
  - for the lack of detection of structural similarities, the class hierarchy was not modified in deep
  - several files with more than 20% of duplication, among these: 3 present more than the 30% of duplication.

- Similarities were hard to be identified
  - structural similarities and duplications should draw the reengineering process
  - a manual reengineering with a greater precision was impossible with only 6 MM of effort
TREND Pre-Processing

- Pre-processing of old MWB and VM Files

<table>
<thead>
<tr>
<th>Algorithm</th>
<th>SI</th>
</tr>
</thead>
<tbody>
<tr>
<td>No Preprocessing</td>
<td>6.88</td>
</tr>
<tr>
<td>SPA</td>
<td>8.38</td>
</tr>
<tr>
<td>SFA</td>
<td>10.63</td>
</tr>
<tr>
<td>SPA+SFA</td>
<td>13.50</td>
</tr>
<tr>
<td>SPA+SFA+VSA</td>
<td>17.25</td>
</tr>
</tbody>
</table>

- *SI* is the System Duplication Index (mean of IID matrices)
- data obtained for the estimation at file level

- **SPA+SFA+VSA** was the best solution for preparing the duplication analysis as confirmed by the experts considering the values estimated by the single modules
File Duplication Analysis, Old Versions

- Only files with HM > 27% have been reported
- Cxx and Hxx files were considered

<table>
<thead>
<tr>
<th>FILE1</th>
<th>FILE2</th>
<th>IIDXY</th>
<th>IIDXY_S</th>
<th>HM</th>
</tr>
</thead>
<tbody>
<tr>
<td>MO_list.hxx</td>
<td>MO_VM.cxx</td>
<td>85</td>
<td>83</td>
<td>70</td>
</tr>
<tr>
<td>MO_VDialog.cxx</td>
<td>MO_gpro.cxx</td>
<td>59</td>
<td>39</td>
<td>38</td>
</tr>
<tr>
<td>MO_list.hxx</td>
<td>MO_MWB.cxx</td>
<td>61</td>
<td>37</td>
<td>37</td>
</tr>
<tr>
<td>MO_gpro.cxx</td>
<td>MO_VDialog.cxx</td>
<td>56</td>
<td>36</td>
<td>35</td>
</tr>
<tr>
<td>MO_list.hxx</td>
<td>MO_VDialog.cxx</td>
<td>60</td>
<td>35</td>
<td>35</td>
</tr>
<tr>
<td>MO_MWB.cxx</td>
<td>MO_VM.cxx</td>
<td>56</td>
<td>31</td>
<td>33</td>
</tr>
<tr>
<td>MO_MWB.hxx</td>
<td>MO_VM.hxx</td>
<td>54</td>
<td>34</td>
<td>33</td>
</tr>
<tr>
<td>MO_VM.cxx</td>
<td>MO_MWB.cxx</td>
<td>51</td>
<td>24</td>
<td>32</td>
</tr>
<tr>
<td>MO_list.hxx</td>
<td>MO_gpro.cxx</td>
<td>57</td>
<td>28</td>
<td>30</td>
</tr>
<tr>
<td>MO_VDialog.cxx</td>
<td>MO_VM.cxx</td>
<td>52</td>
<td>29</td>
<td>30</td>
</tr>
</tbody>
</table>

- This analysis confirmed the work performed by the experts
  - deletion of MO_list.hxx
  - moving part of MO_MWB.cxx and MO_VM.cxx into LIB
File vs Class Level Analysis

- File level analysis is too coarse for reasoning on classes and class hierarchy
- In many cases,
  - ♣ Hxx files contain more than one class (negative aspect)
  - ♣ Cxx files contain a large number of methods
- Class level analysis
  - ♣ structural analysis of classes
  - ♣ duplication analysis of class definition (non useful)
  - ♣ duplication analysis on class methods

- These analysis produce complementary information
  - ♣ In terms of attributes’ types two classes may similar or even identical structure, but
  - ♣ their functional part, methods, may confirm or not the similarity.
Class Structural Analysis of the Old Versions

- **Similar classes**
- **Self Similarity in VM**
- **Self Similarity in MWB**
- **Similarity analysis VM-MWB**

On 76 classes of the old systems, 5776 values, these are those with HMM >= 80%

<table>
<thead>
<tr>
<th>Class 1 (FILE)</th>
<th>Class 2 (FILE)</th>
<th>NAL1</th>
<th>NAL2</th>
<th>NALST</th>
<th>HMM</th>
</tr>
</thead>
<tbody>
<tr>
<td>New_Sys_Metric (MO_VM.hxx)</td>
<td>New_Class_Metric (MO_VM.hxx)</td>
<td>74</td>
<td>75</td>
<td>74</td>
<td>99</td>
</tr>
<tr>
<td>Function (MO_VM.hxx)</td>
<td>Class (MO_VM.hxx)</td>
<td>13</td>
<td>12</td>
<td>11</td>
<td>88</td>
</tr>
<tr>
<td>System_Custom_Metric_Parser (MO_VM.hxx)</td>
<td>Class_Custom_Metric_Parser (MO_VM.hxx)</td>
<td>23</td>
<td>31</td>
<td>23</td>
<td>85</td>
</tr>
<tr>
<td>Function (MO_VM.hxx)</td>
<td>Method (MO_VM.hxx)</td>
<td>13</td>
<td>18</td>
<td>13</td>
<td>84</td>
</tr>
<tr>
<td>Method (MO_VM.hxx)</td>
<td>Class (MO_VM.hxx)</td>
<td>18</td>
<td>12</td>
<td>12</td>
<td>80</td>
</tr>
<tr>
<td>Contenitore (MO_MWB.hxx)</td>
<td>Contenitore_Value (MO_MWB.hxx)</td>
<td>2</td>
<td>3</td>
<td>2</td>
<td>80</td>
</tr>
<tr>
<td>Contenitore_Value (MO_MWB.hxx)</td>
<td>Global (MO_MWB.hxx)</td>
<td>3</td>
<td>2</td>
<td>2</td>
<td>80</td>
</tr>
<tr>
<td>Contenitore_Value (MO_MWB.hxx)</td>
<td>Metriche (MO_MWB.hxx)</td>
<td>3</td>
<td>2</td>
<td>2</td>
<td>80</td>
</tr>
<tr>
<td>Contenitore_Value (MO_MWB.hxx)</td>
<td>VMDialog (MO_MWB.hxx)</td>
<td>3</td>
<td>2</td>
<td>2</td>
<td>80</td>
</tr>
<tr>
<td>PlotDialog (MO_MWB.hxx)</td>
<td>Variable (MO_VM.hxx)</td>
<td>5</td>
<td>4</td>
<td>4</td>
<td>89</td>
</tr>
<tr>
<td>Attributo (MO_MWB.hxx)</td>
<td>Variable (MO_VM.hxx)</td>
<td>3</td>
<td>4</td>
<td>3</td>
<td>86</td>
</tr>
<tr>
<td>Method (MO_MWB.hxx)</td>
<td>View (MO_VM.hxx)</td>
<td>6</td>
<td>6</td>
<td>5</td>
<td>83</td>
</tr>
<tr>
<td>Attributo (MO_MWB.hxx)</td>
<td>Parent (MO_VM.hxx)</td>
<td>3</td>
<td>2</td>
<td>2</td>
<td>80</td>
</tr>
<tr>
<td>Contenitore_Value (MO_MWB.hxx)</td>
<td>Global_Variable (MO_VM.hxx)</td>
<td>3</td>
<td>2</td>
<td>2</td>
<td>80</td>
</tr>
<tr>
<td>Info (MO_MWB.hxx)</td>
<td>Parent (MO_VM.hxx)</td>
<td>3</td>
<td>2</td>
<td>2</td>
<td>80</td>
</tr>
<tr>
<td>InfoDialog (MO_MWB.hxx)</td>
<td>Variable (MO_VM.hxx)</td>
<td>6</td>
<td>4</td>
<td>4</td>
<td>80</td>
</tr>
<tr>
<td>LISTA (MO_MWB.hxx)</td>
<td>Container (MO_VM.hxx)</td>
<td>2</td>
<td>3</td>
<td>2</td>
<td>80</td>
</tr>
<tr>
<td>Method (MO_MWB.hxx)</td>
<td>Variable (MO_VM.hxx)</td>
<td>6</td>
<td>4</td>
<td>4</td>
<td>80</td>
</tr>
</tbody>
</table>
Class Functional Analysis of Old Versions

- MWB and VM class methods with duplications (IID>60%)
- VM self duplication class methods (IID >70%)
- MWB self duplication class methods (IID>50%)

<table>
<thead>
<tr>
<th>Class1</th>
<th>Class2</th>
<th>NL1</th>
<th>NL2</th>
<th>IID_S12</th>
<th>IID_S21</th>
<th>HM</th>
</tr>
</thead>
<tbody>
<tr>
<td>New_Class_Metric</td>
<td>New_Sys_Metric</td>
<td>824</td>
<td>776</td>
<td>91</td>
<td>96</td>
<td>94</td>
</tr>
<tr>
<td>Class_Custom_Metric_Parser</td>
<td>System_Custom_Metric_Parser</td>
<td>200</td>
<td>147</td>
<td>74</td>
<td>98</td>
<td>84</td>
</tr>
<tr>
<td>Function</td>
<td>Method</td>
<td>194</td>
<td>228</td>
<td>74</td>
<td>78</td>
<td>75</td>
</tr>
<tr>
<td>Attribute</td>
<td>Variable</td>
<td>110</td>
<td>60</td>
<td>76</td>
<td>67</td>
<td>74</td>
</tr>
<tr>
<td>Class</td>
<td>Function</td>
<td>189</td>
<td>194</td>
<td>74</td>
<td>76</td>
<td>73</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Class1</th>
<th>Class2</th>
<th>NL1</th>
<th>NL2</th>
<th>IID_S12</th>
<th>IID_S21</th>
<th>HM</th>
</tr>
</thead>
<tbody>
<tr>
<td>UkmDialog</td>
<td>VMDialog</td>
<td>139</td>
<td>114</td>
<td>81</td>
<td>71</td>
<td></td>
</tr>
<tr>
<td>CustomMetric</td>
<td>MetricMember</td>
<td>105</td>
<td>198</td>
<td>58</td>
<td>59</td>
<td></td>
</tr>
<tr>
<td>Single_Metric</td>
<td>Value</td>
<td>34</td>
<td>20</td>
<td>65</td>
<td>51</td>
<td></td>
</tr>
</tbody>
</table>
Summary: Class Analysis of the Old Versions

- The selection of classes presenting a structural similarity and/or functional duplication has allowed to identify the most heavy duplicated classes.
- This approach allowed to detect and analyse all relevant duplicated classes in the 30% of time needed to “manual” operation (including the results analysis).
- Great part of classes with HM>50% have been also manipulated by the team during manual reengineering.
- VM presented a stronger self duplication than MWB.
- Common classes have been detected and moved into the LIB.

<table>
<thead>
<tr>
<th>Operation to Perform</th>
<th>Number of Classes</th>
</tr>
</thead>
<tbody>
<tr>
<td>Deletion</td>
<td>20</td>
</tr>
<tr>
<td>Moving in the library</td>
<td>20</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>System</th>
<th>Nfile</th>
<th>NCL</th>
<th>TNML</th>
<th>TNAL</th>
</tr>
</thead>
<tbody>
<tr>
<td>MWB/VM old</td>
<td>7</td>
<td>76</td>
<td>589</td>
<td>674</td>
</tr>
<tr>
<td>MWB/VM new</td>
<td>10</td>
<td>62</td>
<td>530</td>
<td>640</td>
</tr>
<tr>
<td>MWB/VM TREND</td>
<td>112</td>
<td>56</td>
<td>505</td>
<td>623</td>
</tr>
</tbody>
</table>
Method Analysis of the Old Versions

- Identification of similar methods
  - in the same classes
  - in different classes
  - structural reasoning about the method parameters
  - flow chart analysis
  - analysis based on flow chart metrics

- reasoning about the distribution of similar methods along the class hierarchy

- Definition of guidelines for manipulating methods
Reengineering Process Analysis, **File Level**

- To Assess the performed manual Reengineering

- Analysis of code movements

- Old systems have been compared with the new versions, main files

- E.g.:
  - MN_Metric_Dialog.cxx/hxx are quite completely new.
  - MN_sys_dati.cxx derives its code from MO_MWB.cxx and MO_VM.cxx

<table>
<thead>
<tr>
<th></th>
<th>MO_gpro.cxx</th>
<th>MO_list.hxx</th>
<th>MO_MWB.cxx</th>
<th>MO_MWB.hxx</th>
<th>MO_VDialog.hxx</th>
<th>MO_VM.cxx</th>
<th>MO_VM.hxx</th>
</tr>
</thead>
<tbody>
<tr>
<td>MN_gpro.cxx</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>MN_MetricDialog.cxx</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>MN_MetricDialog.hxx</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>MN_MWB.cxx</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>MN_MWB.hxx</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>MN_sys_dati.cxx</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>MN_sys_dati.hxx</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>MN_VDialog.cxx</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>MN_VM.cxx</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>MN_VM.hxx</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

HM between 1% and 20%
HM between 21% and 50%
HM between 51% and 80%
HM between 81% and 100%
Reengineering Process Analysis, class level

- New Classes having light columns have been created
- Old classes having more than one dark box highlight still present duplications
- The general spreading of colour gives an idea of the work performed by the reengineering team
Some additional data

- Support for the reengineering process
- Support for assessing the reengineering process
- Strong reduction of time analysis
  Manual to semiautomatic:
  - ♠ 1 MM instead of 8 days
  - ♠ A lot of manually non detected duplications

<table>
<thead>
<tr>
<th>MWB/VM</th>
<th>Files</th>
<th>Classes</th>
<th>Methods</th>
</tr>
</thead>
<tbody>
<tr>
<td>Number</td>
<td>7</td>
<td>76</td>
<td>589</td>
</tr>
<tr>
<td>#comparisons</td>
<td>21</td>
<td>2.850</td>
<td>173.166</td>
</tr>
<tr>
<td>Time CASH alg</td>
<td>3 s</td>
<td>140 s</td>
<td>530 s</td>
</tr>
<tr>
<td>Number of Lines</td>
<td>11.149</td>
<td>10.380</td>
<td>10.380</td>
</tr>
<tr>
<td>Total Byes</td>
<td>389.921</td>
<td>248.331</td>
<td>248.331</td>
</tr>
</tbody>
</table>
Conclusions

● A Method and its adoption for
  ♦ duplication detection and analysis
  ♦ similarity detection and analysis
  ♦ it can be used on different languages and cases

● Duplication analysis of object-oriented systems
  ➔ file, class, and method levels
  ♦ file level analysis is not enough to get conclusions
  ♦ simple duplication metrics and tools are not effective, HM

● The tools defined includes
  ♦ algorithms for duplication and similarity
  ♦ specific metrics for the process
  ♦ suitable visualisation tools
  ♦ code reorganisation tool
Detailed Estimation Costs

By knowing the effort of maintenance it is possible to estimate the costs of: duplication, deletion and addition.

\[
dup[i] = \frac{NLF_2[i] \times IID_2[i]}{100}
\]

Number of reused/duplicated lines

\[
ew[i] = NLF_2[i] - dup[i]
\]

Number of new/added lines

\[
del[i] = NLF_1[i] - dup[i]
\]

Number deleted lines

\[
E_M[i] = E_{\text{Effort Measure}} \text{ for each class couple } i \text{ of class } C_1[i], C_2[i].
\]

Multilinear Regression to estimate \(a, b, c, d\)

\[
E_M[i] = a \times dup[i] + b \times new[i] + c \times del[i] + d
\]
Detailed Estimation Costs \textit{a real case}

1) multilinear regression analysis

<table>
<thead>
<tr>
<th>Metric</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>R</td>
<td>0.858</td>
</tr>
<tr>
<td>R-squared</td>
<td>0.736</td>
</tr>
<tr>
<td>stderr</td>
<td>47.9</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Coeffi</th>
<th>p-val</th>
</tr>
</thead>
<tbody>
<tr>
<td>(d)</td>
<td>10.68 (0.0086)</td>
</tr>
<tr>
<td>(a)</td>
<td>0.245 (3.86E-18)</td>
</tr>
<tr>
<td>(b)</td>
<td>0.197 (4.34E-15)</td>
</tr>
<tr>
<td>(c)</td>
<td>0.463 (2.01E-16)</td>
</tr>
</tbody>
</table>

\(3)\) Correlation Analysis

\textit{Estimated Effort} = a* DUP + b*NEW + c*DEL + d *N

DUP = a*\(\sum\) dup[i] : effort of code reuse
NEW = b*\(\sum\) new[i] : effort of code addition
DEL = c*\(\sum\) del[i] : effort of code deletion
D = d*N : fixed cost of code analysis