Using Structured Text Source Code Metrics and Artificial Neural Networks to Predict Change Proneness at Code Tab and Program Organization Level

Lov Kumar\textsuperscript{1} \hspace{1cm} Ashish Sureka\textsuperscript{2}

\textsuperscript{1}NIT Rourkela, India
Email: lovkumar505@gmail.com

\textsuperscript{2}ABB India, India
Email: ashish.sureka@in.abb.com
# Table of Contents

1. **Research Motivation and Aim**
   - Introduction - Industrial Automation
   - Research Objectives and Focus

2. **Related Work and Research Contributions**
   - Literature Survey
   - Research Contributions

3. **Experimental Dataset**

4. **Research Methodology**
   - High Level Architecture
   - Proposed Source Code Metrics

5. **Code Metrics - Statistics, Correlation**
   - Descriptive Statistics
   - Correlations Between 10 Source Code Metrics
   - Data Annotation using Fuzzy Clustering

6. **Experimental Results**
   - Feature Extraction and Selection
   - ANN Training and Performance Evaluation
   - t-Test and Classification Methods

7. **Conclusion**
PLC (Programmable Logic Controller) Programming

The IEC (International Electrotechnical Commission) 61131-3 international standard provides guidelines for PLC (Programmable Logic Controller) programming and is accepted as a standard by PLC manufacturers [3][9].

ST (Structured Text) is a high-level powerful language - probably the most widely used controller programming language in the industrial automation engineering domain [3][9].

Several characteristics of ST is different than that of general purpose programming languages as the primary purpose of a PLC is to control an industrial process [12].
Industrial Automation Applications

**Source code metrics** for computing the structural complexity of PLC applications and control systems developed using **ST** are different than non-industrial automation applications developed using general purpose programming languages.

Area of **change proneness prediction** using source code metrics for control systems in automation engineering domain is unexplored.

Lack of availability of open-source automation engineering projects and **lack of research studies from industries** on commercial data.
Table of Contents

1 Research Motivation and Aim
   • Introduction - Industrial Automation
   • Research Objectives and Focus

2 Related Work and Research Contributions
   • Literature Survey
   • Research Contributions

3 Experimental Dataset

4 Research Methodology
   • High Level Architecture
   • Proposed Source Code Metrics

5 Code Metrics - Statistics, Correlation
   • Descriptive Statistics
   • Correlations Between 10 Source Code Metrics
   • Data Annotation using Fuzzy Clustering

6 Experimental Results
   • Feature Extraction and Selection
   • ANN Training and Performance Evaluation
   • t-Test and Classification Methods

7 Conclusion
Research Aim

[1] To conduct experiments on two real-world, large, complex, matured, active and diverse projects at an automation engineering company.

[2] To investigate the relation between 10 source code metrics between themselves and with change-proneness at two level of granularity (Code Tab and Program Organization Unit).

[3] To build change proneness prediction model based on Artificial Neural Networks (ANN).

Love Kumar, Ashish Sureka

Using Structured Text Source Code Metrics and Artificial Neural Networks to Predict Change Proneness at Code Tab and Program Organization Level

Table of Contents

1 Research Motivation and Aim
   • Introduction - Industrial Automation
   • Research Objectives and Focus

2 Related Work and Research Contributions
   • Literature Survey
   • Research Contributions

3 Experimental Dataset

4 Research Methodology
   • High Level Architecture
   • Proposed Source Code Metrics

5 Code Metrics - Statistics, Correlation
   • Descriptive Statistics
   • Correlations Between 10 Source Code Metrics
   • Data Annotation using Fuzzy Clustering

6 Experimental Results
   • Feature Extraction and Selection
   • ANN Training and Performance Evaluation
   • t-Test and Classification Methods

7 Conclusion
Code Analysis for PLC Languages

Kumar et al. [5]
Kumar et al. present source code level metrics to measure size, vocabulary, cognitive complexity and testing complexity of Ladder Diagram (LD) which is a visual PLC programming language [5]

Nair et al. [8]
Nair et al. presents a methodology to define metrics for IEC 61131-3 domain specific languages. Using their proposed methodology, they define a set of product metrics that can be used for managing the software project development using PLC languages [8]
Prahofer et al. [10]

Prahofer et al. mention that static code analysis tools are rare in the domain of PLC programming and they present an approach and tool support for static code analysis of PLC programs [10].
Lu et al. [6]
Lu et al. conduct experiments on 102 Java systems to investigate the ability of 62 Object-Oriented metrics to predict change-proneness [6].

Koru et al. [4]
Koru et al. conduct experiments on two open-source projects (KOffice and Mozilla) and identified and characterized the change-prone classes in these two products by producing tree-based models [4].

Romano et al. [11]
Romano et al. investigate the extent to which existing source code metrics can be used for predicting change-prone Java interfaces [11].
# Table of Contents

1. Research Motivation and Aim  
   - Introduction - Industrial Automation  
   - Research Objectives and Focus  

2. Related Work and Research Contributions  
   - Literature Survey  
   - Research Contributions

3. Experimental Dataset

4. Research Methodology  
   - High Level Architecture  
   - Proposed Source Code Metrics

5. Code Metrics - Statistics, Correlation  
   - Descriptive Statistics  
   - Correlations Between 10 Source Code Metrics  
   - Data Annotation using Fuzzy Clustering

6. Experimental Results  
   - Feature Extraction and Selection  
   - ANN Training and Performance Evaluation  
   - t-Test and Classification Methods

7. Conclusion

Lov Kumar, Ashish Sureka  
Using Structured Text Source Code Metrics and Artificial Neural
Novel and Unique Contributions

[1] First study on change-proneness prediction in the domain of automation engineering for PLC programming languages

[2] We conduct experiments on two real-world, large and complex software developed and maintained at an industrial automation engineering company

[3] We define and implement 10 source code metrics for Structured Text

[4] Investigate the relationship between source code metrics and change-proneness
Novel and Unique Contributions

[5] Examine the impact of PCA and rough set analysis feature extraction and selection techniques

[6] Apply artificial neural networks using three different training algorithms to build statistical models for predicting change proneness
### Experimental Dataset used in Our Study

<table>
<thead>
<tr>
<th></th>
<th>Version 1</th>
<th></th>
<th>Version 2</th>
<th></th>
<th>Common</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>CT</td>
<td>POU</td>
<td>CT</td>
<td>POU</td>
<td>CT</td>
</tr>
<tr>
<td><strong>Project 1</strong></td>
<td>214</td>
<td>82</td>
<td>240</td>
<td>82</td>
<td>158</td>
</tr>
<tr>
<td><strong>Project 2</strong></td>
<td>293</td>
<td>104</td>
<td>344</td>
<td>104</td>
<td>209</td>
</tr>
</tbody>
</table>

**Experimental Dataset** [CT: Number of Code Tabs, POU: Number of Program Organization Units]

**Program Organization Unit (POU)** are software units within an application
POUs are independent units or building blocks of the programming system.

A POU within the programming system can contain multiple Code Tabs (input and output variable declarations and control logic) [3][9]

We take two versions of two real world projects in our organization

POUs can be further classified into three types: Functions (FUN), Function Blocks (FB) and Programs (PROG) [7]

We conduct experiments on dataset belonging to two projects of different sizes to test the generalizability of our approach
# Table of Contents

1. **Research Motivation and Aim**
   - Introduction - Industrial Automation
   - Research Objectives and Focus

2. **Related Work and Research Contributions**
   - Literature Survey
   - Research Contributions

3. **Experimental Dataset**

4. **Research Methodology**
   - High Level Architecture
   - Proposed Source Code Metrics

5. **Code Metrics - Statistics, Correlation**
   - Descriptive Statistics
   - Correlations Between 10 Source Code Metrics
   - Data Annotation using Fuzzy Clustering

6. **Experimental Results**
   - Feature Extraction and Selection
   - ANN Training and Performance Evaluation
   - t-Test and Classification Methods

7. **Conclusion**
Lov Kumar, Ashish Sureka

Using Structured Text Source Code Metrics and Artificial Neural Networks to Predict Change Proneness at Code Tab and Program Organization Level
Source code metrics serves as the predictor or independent variables.

We apply a data-driven fuzzy clustering based algorithm to annotate the data into three categories: high, medium and low change-proneness.

We conduct experiments with three different set of source code metrics.

We apply Artificial Neural Network (ANN) with three different training algorithms i.e., Gradient Descent method (GD), Quasi-Newton method (NM), and Levenberg-Marquardt (LM).
We use 10-fold cross validation to create different partitions of training and testing data and generalize the result of our analysis.

We compute the predictive accuracy of various models using confusion matrix and then applying t-test analysis to identify the most accurate model.
Table of Contents

1. Research Motivation and Aim
   - Introduction - Industrial Automation
   - Research Objectives and Focus

2. Related Work and Research Contributions
   - Literature Survey
   - Research Contributions

3. Experimental Dataset

4. Research Methodology
   - High Level Architecture
   - Proposed Source Code Metrics

5. Code Metrics - Statistics, Correlation
   - Descriptive Statistics
   - Correlations Between 10 Source Code Metrics
   - Data Annotation using Fuzzy Clustering

6. Experimental Results
   - Feature Extraction and Selection
   - ANN Training and Performance Evaluation
   - t-Test and Classification Methods

7. Conclusion
We propose, define and implement 10 source code metrics. Except the Cognitive Complexity (CC) and Testing Complexity (TC), rest all metrics are same for the POU and Code Tab level.

Size

Size: We define size as the number of LOC (Lines of Code) metric (excluding the comments) which is equal to the number of executable statements in the Structured Text program.
10 source code metrics

Vocabulary

**Vocabulary:** We define vocabulary as the total number of distinct operators and operands used in a given program.

%\text{CMT}

**%CMT:** It measures the percentage of comment in source (% of comments in LOC)

Program Length

**Program Length:** We define program length as the total number of operators and operands used in a given ST program.
Calculated Program Length (Cproglen)

Calculated Program Length (Cproglen) We define the Cproglen of a ST program using the following equation:

\[ C_{proglen} = \eta_1 \log \eta_1 + \eta_2 \log \eta_2 \]  \hspace{1cm} (1)

where \( \eta_1 \) and \( \eta_1 \) represents the total number of distinct operators and operands respectively.
10 source code metrics

<table>
<thead>
<tr>
<th>Volume</th>
</tr>
</thead>
</table>

**Volume**: We define the Volume of a ST program using the following equation:

\[
Volume = (N_1 + N_2) \log(\eta_1 + \eta_2)
\]  

(2)

where \(N_1\), and \(N_2\) represents the total number of operators and operands in a given text.

<table>
<thead>
<tr>
<th>Difficulty</th>
</tr>
</thead>
</table>

**Difficulty**: We define the Difficulty of a ST program using the following equation:

\[
Difficulty = \frac{\eta_1}{2} \times \frac{N_2}{\eta_2}
\]  

(3)
Effort

Effort of a structured text program is computed as the product of volume and difficulty.

Complexity (CC) of a given POU

How easy or difficult it is to understand and comprehend the POU:

\[
CC_{POU} = (\eta_1 + \eta_2) \times \left( \sum_{i=1}^{n} CW_{CT_i} + \sum_{j=1}^{m} CW_{AT_j} + \sum_{k=1}^{o} CW_{IH_k} \right) \tag{4}
\]

where \( \eta_1 \) and \( \eta_2 \) are the number of distinct operator and operands in the POU. \( CW_{CT} \), \( CW_{AT} \), and \( CW_{IH} \) represents the cognitive weight of the Code Tab, Attribute, and Inheritance Level respectively.
10 source code metrics

**Code Tab Cognitive Weight**

Code Tab Cognitive Weight ($CW_{CT}$) is used to calculate the complexity of a Code Tab in a POU. It is computed using following equation:

$$CW_{CT} = \sum_{i=1}^{P} W_i$$  \hspace{1cm} (5)

where $P$ represents the number of basic control structures and $W_i$ represents the cognitive weight of $i^{th}$ basic control structure.
Attribute Cognitive Weight (\(CW_{AT}\)) is used to calculate the complexity of attributes used in the POU. It is calculated using the following equation:

\[
CW_{AT} = NPDT \times W_{PDT} + NUDT \times W_{UDT}
\]

(6)

where \(NPDT\), and \(NUDT\) represents the number of predefined and user defined data types in the POU and \(W_{PDT}\) and \(W_{UDT}\) represents the weight of predefined and user defined data types respectively.
10 source code metrics

Inheritance Level Cognitive Weight

Inheritance Level Cognitive Weight \((CW_{IH})\) is used to calculate the complexity of inheritance level of POU. It is calculated using the following equation:

\[
CW_{IH} = DIT \times NOA \times CL
\]  

(7)

where DIT, NOA, and CL represents the depth of the inheritance tree, number of attribute used for inheritance and cognitive weight of the inheritance level.
10 source code metrics

Testing Complexity

**Testing Complexity**: Testing Complexity (TC) of a POU is defined as the number of test cases required to test the program. We define Testing Complexity (TC) as a measure of the number of all possible control flows in POU. TC at the POU level is defined using the following equation:

\[
TC_{POU} = \sum_{i=1}^{n} TC_{CTi} - n
\]  

(8)

where \( TC_{POU} \) and \( TC_{CT} \) represents the testing complexity of POU and Code Tab respectively and \( n \) represents the number of Code Tabs in the given POU.
Lov Kumar, Ashish Sureka

Using Structured Text Source Code Metrics and Artificial Neural

Table of Contents

1. Research Motivation and Aim
   • Introduction - Industrial Automation
   • Research Objectives and Focus

2. Related Work and Research Contributions
   • Literature Survey
   • Research Contributions

3. Experimental Dataset

4. Research Methodology
   • High Level Architecture
   • Proposed Source Code Metrics

5. Code Metrics - Statistics, Correlation
   • Descriptive Statistics
   • Correlations Between 10 Source Code Metrics
   • Data Annotation using Fuzzy Clustering

6. Experimental Results
   • Feature Extraction and Selection
   • ANN Training and Performance Evaluation
   • t-Test and Classification Methods

7. Conclusion
## Project 1 - Descriptive Statistics (CodeTab)

<table>
<thead>
<tr>
<th>Metrics</th>
<th>Code Tab Level</th>
<th>Min.</th>
<th>Max.</th>
<th>Mean</th>
<th>Median</th>
<th>Std Dev.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Size</td>
<td></td>
<td>1</td>
<td>198</td>
<td>16</td>
<td>6</td>
<td>27.46</td>
</tr>
<tr>
<td>ProgLen</td>
<td></td>
<td>4</td>
<td>2752</td>
<td>195.18</td>
<td>83</td>
<td>327.74</td>
</tr>
<tr>
<td>Vocabulary</td>
<td></td>
<td>4</td>
<td>227</td>
<td>37.27</td>
<td>30</td>
<td>30.47</td>
</tr>
<tr>
<td>Cproglen</td>
<td></td>
<td>2.77</td>
<td>1163.79</td>
<td>126.03</td>
<td>82.32</td>
<td>148.12</td>
</tr>
<tr>
<td>Volume</td>
<td></td>
<td>5.55</td>
<td>14929.46</td>
<td>812.74</td>
<td>290.03</td>
<td>1626.02</td>
</tr>
<tr>
<td>Difficulty</td>
<td></td>
<td>1</td>
<td>69.59</td>
<td>12.17</td>
<td>8.42</td>
<td>12.52</td>
</tr>
<tr>
<td>Effort</td>
<td></td>
<td>5.55</td>
<td>941205.24</td>
<td>24044.21</td>
<td>2402.74</td>
<td>88355.21</td>
</tr>
<tr>
<td>CC</td>
<td></td>
<td>4</td>
<td>66057</td>
<td>1804.26</td>
<td>256</td>
<td>6577.14</td>
</tr>
<tr>
<td>TC</td>
<td></td>
<td>1</td>
<td>42</td>
<td>3.49</td>
<td>2</td>
<td>5.31</td>
</tr>
<tr>
<td>% Comment</td>
<td></td>
<td>0</td>
<td>0.55</td>
<td>0.03</td>
<td>0.01</td>
<td>0.06</td>
</tr>
</tbody>
</table>
## Project 2 - Descriptive Statistics (CodeTab)

<table>
<thead>
<tr>
<th>Metrics</th>
<th>Code Tab Level</th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Min.</td>
<td>Max.</td>
<td>Mean</td>
<td>Median</td>
<td>Std Dev.</td>
</tr>
<tr>
<td>Size</td>
<td>1</td>
<td>216</td>
<td>23.29</td>
<td>10</td>
<td>35.18</td>
</tr>
<tr>
<td>ProgLen</td>
<td>4</td>
<td>3791</td>
<td>350</td>
<td>114</td>
<td>571.7</td>
</tr>
<tr>
<td>Vocabulary</td>
<td>4</td>
<td>245</td>
<td>47.66</td>
<td>34</td>
<td>42.96</td>
</tr>
<tr>
<td>Cproglen</td>
<td>2.77</td>
<td>1310.45</td>
<td>178.77</td>
<td>100.88</td>
<td>221.86</td>
</tr>
<tr>
<td>Volume</td>
<td>5.55</td>
<td>16265.13</td>
<td>1542.44</td>
<td>395.09</td>
<td>2744.13</td>
</tr>
<tr>
<td>Difficulty</td>
<td>1</td>
<td>186.55</td>
<td>17.38</td>
<td>9.39</td>
<td>25.39</td>
</tr>
<tr>
<td>Effort</td>
<td>5.55</td>
<td>3034260.33</td>
<td>75949.77</td>
<td>3634.67</td>
<td>284794.55</td>
</tr>
<tr>
<td>CC</td>
<td>4</td>
<td>104976</td>
<td>4287.78</td>
<td>378</td>
<td>12755.03</td>
</tr>
<tr>
<td>TC</td>
<td>1</td>
<td>42</td>
<td>3.56</td>
<td>2</td>
<td>5.21</td>
</tr>
<tr>
<td>% Comment</td>
<td>0</td>
<td>0.55</td>
<td>0.04</td>
<td>0.02</td>
<td>0.06</td>
</tr>
</tbody>
</table>
The descriptive statistics describes and characterizes the features of the two versions of the two projects.

We observe enough variance and dispersion in majority of the variable values.

We observe wide variance in various source code complexity metrics within the same project across Code Tabs and POUs.

The variance in code complexity metrics shows that some modules or units are more complex than others.
### Project 1 (Upper Triangle) and Project 2 (Lower triangle)

#### CodeTab Level

<table>
<thead>
<tr>
<th></th>
<th>SZ</th>
<th>PL</th>
<th>VC</th>
<th>CPL</th>
<th>VOL</th>
<th>DIF</th>
<th>EFF</th>
<th>CC</th>
<th>TC</th>
<th>PC</th>
</tr>
</thead>
<tbody>
<tr>
<td>PC</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>TC</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>CC</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>EFF</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>DIF</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>VOL</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>CPL</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>VC</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>PL</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>SZ</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

#### Descriptive Statistics

- Correlations Between 10 Source Code Metrics
- Data Annotation using Fuzzy Clustering

Lov Kumar, Ashish Sureka

Using Structured Text Source Code Metrics and Artificial Neural Network to Predict Change Proneness at Code Tab and Program Organization Level
Using Structured Text Source Code Metrics and Artificial Neural Networks to Predict Change Proneness at Code Tab and Program Organization Level
We compute the dependency between 10 code metrics using Pearson Correlation Coefficient.

A black circle denotes a $r$ value between 0.7 and 1.0 indicating a strong positive linear relationship.

A white circle denotes a $r$ value between 0.3 and 0.7 indicating a weak positive linear relationship.

An empty cell indicates no linear relationship. We did not find instances of negative relationship.
Table of Contents

1. Research Motivation and Aim
   - Introduction - Industrial Automation
   - Research Objectives and Focus

2. Related Work and Research Contributions
   - Literature Survey
   - Research Contributions

3. Experimental Dataset

4. Research Methodology
   - High Level Architecture
   - Proposed Source Code Metrics

5. Code Metrics - Statistics, Correlation
   - Descriptive Statistics
   - Correlations Between 10 Source Code Metrics
   - Data Annotation using Fuzzy Clustering

6. Experimental Results
   - Feature Extraction and Selection
   - ANN Training and Performance Evaluation
   - t-Test and Classification Methods

7. Conclusion
Fuzzy Clustering Technique

We define change proneness of a POU or a Code Tab into three categories: High (H), Medium (M) and Low (L)

We compute the number of changed lines of code between two versions of the system at the Code Tab and POU level both the projects

Instead of arbitrarily defining a fixed threshold for the number of lines of code to categorize each Code Tab or POU into H, M or L, we apply fuzzy clustering technique (data driven) to automatically derive or determine the threshold and classification of each unit into categories
Fuzzy Clustering Technique

Table: Fuzzy Cluster Centers [Lines Changed] of POU

| Proneness | Project 1 | | Project 2 | | |
|-----------|-----------|-----------|-----------|-----------|
|           | CT Level  | POU Level | CT Level  | POU Level |
| **C1**    | 4.93      | 16.02     | 8.45      | 23.04     |
| **C2**    | 60.30     | 205.42    | 97.87     | 331.91    |
| **C3**    | 172.03    | 505.45    | 244.49    | 788.43    |
Table of Contents

1 Research Motivation and Aim
   • Introduction - Industrial Automation
   • Research Objectives and Focus

2 Related Work and Research Contributions
   • Literature Survey
   • Research Contributions

3 Experimental Dataset

4 Research Methodology
   • High Level Architecture
   • Proposed Source Code Metrics

5 Code Metrics - Statistics, Correlation
   • Descriptive Statistics
   • Correlations Between 10 Source Code Metrics
   • Data Annotation using Fuzzy Clustering

6 Experimental Results
   • Feature Extraction and Selection
   • ANN Training and Performance Evaluation
   • t-Test and Classification Methods

7 Conclusion

Lov Kumar, Ashish Sureka
Using Structured Text Source Code Metrics and Artificial Neural
## Descriptive Statistics of Code Tabs and POUs

<table>
<thead>
<tr>
<th>CP</th>
<th>Project 1</th>
<th>Project 2</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Code Tab Level</td>
<td>POU Level</td>
</tr>
<tr>
<td></td>
<td>No.</td>
<td>%</td>
</tr>
<tr>
<td>Low (L)</td>
<td>127</td>
<td>80.89</td>
</tr>
<tr>
<td>Medium (M)</td>
<td>21</td>
<td>13.38</td>
</tr>
<tr>
<td>High (H)</td>
<td>9</td>
<td>5.73</td>
</tr>
</tbody>
</table>

Table shows the **descriptive statistics** of Code Tabs and POUs in terms of categorization into High, Medium and Low lines changed.

Table reveals that there are 127, 21 and 9 Code Tabs belonging to the Low, Medium and High category.
### Principal Component Analysis (PCA)

<table>
<thead>
<tr>
<th>Feature</th>
<th>CodeTab Level</th>
<th>POU Level</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>PC1</td>
<td>PC2</td>
</tr>
<tr>
<td>Size</td>
<td>0.33</td>
<td>0.30</td>
</tr>
<tr>
<td>ProgLen</td>
<td>0.36</td>
<td>-0.13</td>
</tr>
<tr>
<td>vocabulary</td>
<td>0.32</td>
<td>-0.30</td>
</tr>
<tr>
<td>Cproglcen</td>
<td>0.32</td>
<td>-0.35</td>
</tr>
<tr>
<td>Volume</td>
<td>0.36</td>
<td>-0.19</td>
</tr>
<tr>
<td>Difficulty</td>
<td>0.31</td>
<td>0.35</td>
</tr>
<tr>
<td>Effort</td>
<td>0.34</td>
<td>-0.12</td>
</tr>
<tr>
<td>CC</td>
<td>0.33</td>
<td>-0.16</td>
</tr>
<tr>
<td>TC</td>
<td>0.22</td>
<td>0.62</td>
</tr>
<tr>
<td>% cmt</td>
<td>0.23</td>
<td>0.32</td>
</tr>
<tr>
<td>Eigenvalues</td>
<td>6.98</td>
<td>1.36</td>
</tr>
<tr>
<td>% Variance</td>
<td>69.81</td>
<td>13.56</td>
</tr>
<tr>
<td>Cumulative % variance</td>
<td>69.81</td>
<td>83.37</td>
</tr>
</tbody>
</table>
Principal Component Analysis (PCA)

A smaller number of 4 variables accounting for the majority of the variance in the dependent variable

Some of the variables are strongly correlated and some are weakly correlated with the four principal components

**Vocabulary** has a strong correlation with PC4 and % cmt has a strong correlation with PC3. Similarly, we observe a strong correlation between TC and PC2.

The **third principal component** decreases significantly with increase in % cmt and increases significantly with increase in size and TC
Feature Extraction and Selection
ANN Training and Performance Evaluation
t-Test and Classification Methods

Rough Set Theory and Expectation Maximization

**Table:** Selected Set of Source Code Metrics using Rough Set Analysis

<table>
<thead>
<tr>
<th>Project</th>
<th>Level</th>
<th>Source Code Metrics</th>
</tr>
</thead>
<tbody>
<tr>
<td>Project 1</td>
<td>CodeTab</td>
<td>Size, ProgLen, Vocabulary, Cproglen, Difficulty, Effort, TC, %cmt</td>
</tr>
<tr>
<td></td>
<td>POU</td>
<td>Cproglen, Volume, Effort, TC, %cmt</td>
</tr>
<tr>
<td>Project 2</td>
<td>CodeTab</td>
<td>Size, ProgLen, Vocabulary, Cproglen, Volume, Difficulty, TC, %cmt</td>
</tr>
<tr>
<td></td>
<td>POU</td>
<td>Size, Vocabulary, Volume, Difficulty, Effort, CC</td>
</tr>
</tbody>
</table>
Rough Set Theory and Expectation Maximization

Table shows the **subset of source code metrics** selected after applying the rough set theory based technique.

Rough set analysis falls into the category of feature selection wherein we chose the **most informative subset of features** from the original set of features.

Results reveals that the **dimensionality of the attributes** has been reduced from 10 to 5 for Project 1 at POU level. Similarly, the dimensionality has been reduced for Project 2 at both the Code Tab and POU level.
Lov Kumar, Ashish Sureka

Using Structured Text Source Code Metrics and Artificial Neural Networks to Predict Change Proneness at Code Tab and Program Organization Level
## Results Based on Accuracy and F-Measure

<table>
<thead>
<tr>
<th>Project</th>
<th>Level</th>
<th>Classifier</th>
<th>AM</th>
<th>PCA</th>
<th>RSA</th>
<th>AM</th>
<th>PCA</th>
<th>RSA</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Project 1</strong></td>
<td><strong>Code Tab</strong></td>
<td>GD</td>
<td>81.53</td>
<td>80.89</td>
<td>79.62</td>
<td>0.93</td>
<td>0.93</td>
<td>0.92</td>
</tr>
<tr>
<td></td>
<td></td>
<td>NM</td>
<td>82.80</td>
<td>79.62</td>
<td>80.89</td>
<td>0.94</td>
<td>0.92</td>
<td>0.93</td>
</tr>
<tr>
<td></td>
<td></td>
<td>LM</td>
<td>84.71</td>
<td>82.17</td>
<td>84.08</td>
<td>0.94</td>
<td>0.93</td>
<td>0.94</td>
</tr>
<tr>
<td></td>
<td><strong>POU</strong></td>
<td>GD</td>
<td>91.23</td>
<td>85.96</td>
<td>85.96</td>
<td>0.97</td>
<td>0.95</td>
<td>0.95</td>
</tr>
<tr>
<td></td>
<td></td>
<td>NM</td>
<td>89.47</td>
<td>85.96</td>
<td>89.47</td>
<td>0.96</td>
<td>0.95</td>
<td>0.96</td>
</tr>
<tr>
<td></td>
<td></td>
<td>LM</td>
<td>92.98</td>
<td>87.72</td>
<td>91.23</td>
<td>0.98</td>
<td>0.96</td>
<td>0.97</td>
</tr>
<tr>
<td><strong>Project 2</strong></td>
<td><strong>Code Tab</strong></td>
<td>GD</td>
<td>80.38</td>
<td>78.47</td>
<td>80.38</td>
<td>0.92</td>
<td>0.92</td>
<td>0.92</td>
</tr>
<tr>
<td></td>
<td></td>
<td>NM</td>
<td>85.17</td>
<td>77.51</td>
<td>85.17</td>
<td>0.95</td>
<td>0.91</td>
<td>0.95</td>
</tr>
<tr>
<td></td>
<td></td>
<td>LM</td>
<td>89.00</td>
<td>77.99</td>
<td>89.95</td>
<td>0.96</td>
<td>0.91</td>
<td>0.96</td>
</tr>
<tr>
<td></td>
<td><strong>POU</strong></td>
<td>GD</td>
<td>90.28</td>
<td>83.33</td>
<td>91.67</td>
<td>0.97</td>
<td>0.94</td>
<td>0.97</td>
</tr>
<tr>
<td></td>
<td></td>
<td>NM</td>
<td>93.06</td>
<td>88.89</td>
<td>91.67</td>
<td>0.98</td>
<td>0.96</td>
<td>0.97</td>
</tr>
<tr>
<td></td>
<td></td>
<td>LM</td>
<td>95.83</td>
<td>90.28</td>
<td>93.06</td>
<td>0.99</td>
<td>0.97</td>
<td>0.98</td>
</tr>
</tbody>
</table>
Experimental Results

We consider three different subset of metrics as input to design a model for predicting change proneness of PLC programs using ANN with three different type of training algorithm i.e., GD, NM, and LM.

The performance of each prediction model is evaluated in terms of two different performance parameters i.e., accuracy and F-measure.

We observe that the classifier is confusing between the classes M and L indicating areas of improvement.
Project 1 (Code Tab Level)

Lov Kumar, Ashish Sureka

Using Structured Text Source Code Metrics and Artificial Neural Networks to Predict Change Proneness at Code Tab and Program Organization Level
Project 2 (Code Tab Level)

Using Structured Text Source Code Metrics and Artificial Neural Networks to Predict Change Proneness at Code Tab and Program Organization Level
We create box-plot diagrams for each of the experimental results enabling a **visual comparison**.

We apply 10 fold cross validation for all the **combinations** and the accuracy and f-measure metric values are summarized in the box blots.

Each box-plot diagram is **partitioned into three parts**: one for all metrics, one for PCA and one for RSA.

The box-plot diagrams presents **performance of all feature selection methods** within a single diagram.
Table of Contents

1. Research Motivation and Aim
   - Introduction - Industrial Automation
   - Research Objectives and Focus

2. Related Work and Research Contributions
   - Literature Survey
   - Research Contributions

3. Experimental Dataset

4. Research Methodology
   - High Level Architecture
   - Proposed Source Code Metrics

5. Code Metrics - Statistics, Correlation
   - Descriptive Statistics
   - Correlations Between 10 Source Code Metrics
   - Data Annotation using Fuzzy Clustering

6. Experimental Results
   - Feature Extraction and Selection
   - ANN Training and Performance Evaluation
   - t-Test and Classification Methods

7. Conclusion
### Accuracy

<table>
<thead>
<tr>
<th></th>
<th>P-value</th>
<th>Mean Difference</th>
</tr>
</thead>
<tbody>
<tr>
<td>ALL</td>
<td>NaN</td>
<td>0.00</td>
</tr>
<tr>
<td>PCA</td>
<td>0.00</td>
<td>NaN</td>
</tr>
<tr>
<td>RSA</td>
<td>0.06</td>
<td>0.01</td>
</tr>
</tbody>
</table>

### F-Measure

<table>
<thead>
<tr>
<th></th>
<th>P-value</th>
<th>Mean Difference</th>
</tr>
</thead>
<tbody>
<tr>
<td>ALL</td>
<td>NaN</td>
<td>0.05</td>
</tr>
<tr>
<td>PCA</td>
<td>0.00</td>
<td>0.01</td>
</tr>
<tr>
<td>RSA</td>
<td>0.05</td>
<td>NaN</td>
</tr>
</tbody>
</table>
We use **pairwise t-test** to compare the performance of feature selection techniques and classifier training methods.

We use pairwise t-test to investigate if the differences between the multiple classifiers in terms of their accuracy is a co-incidence or random or they are real [1].

We consider ANN with **three different types of training methods** to develop a model to predict change-proneness.

We use **three different subset of metrics** of two different version of PLC projects with two different performance parameters i.e., accuracy and F-Measure.
### Classification Methods

#### Accuracy

<table>
<thead>
<tr>
<th></th>
<th>GD</th>
<th>NM</th>
<th>LM</th>
<th>GD</th>
<th>NM</th>
<th>LM</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>P-value</strong></td>
<td>NaN</td>
<td>0.05</td>
<td>0.00</td>
<td>0.00</td>
<td>-1.66</td>
<td>-4.11</td>
</tr>
<tr>
<td><strong>Mean Difference</strong></td>
<td>4.11</td>
<td>2.44</td>
<td>0.00</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

#### F-Measure

<table>
<thead>
<tr>
<th></th>
<th>GD</th>
<th>NM</th>
<th>LM</th>
<th>GD</th>
<th>NM</th>
<th>LM</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>P-value</strong></td>
<td>NaN</td>
<td>0.04</td>
<td>0.00</td>
<td>0.00</td>
<td>0.00</td>
<td>0.00</td>
</tr>
<tr>
<td><strong>Mean Difference</strong></td>
<td>0.016</td>
<td>0.00</td>
<td>0.00</td>
<td>0.00</td>
<td>0.00</td>
<td>0.00</td>
</tr>
</tbody>
</table>
Pairwise t-test

For each prediction technique a total number of two sets (one for each performance) are used, each with 12 data points \([(2 \text{ feature selection method} + 1 \text{ considering all features}) \times 4 \text{ datasets}]\)

Results reveals that for most of the cases there is a significant difference between different approaches as the p-value is lesser than 0.05.

We observe that the p-value of the GD and NM combination is 0.05. According to the value of mean difference, LM i.e., ANN with LM yields better result compared to other classifiers.
Conclusion and Takeaways

It is possible to accurately predict the change-proneness of Structured Text programs using source code metrics by employing ANNs and PCA and RSA based feature selection techniques.

Artificial Neural Networks with Levenberg-Marquardt training method results in better accuracy (highest median and maximum values of performance parameters) in comparison to other training methods.

It is possible to identify a reduced subset of source code metrics and attributes based on feature extraction and selection technique for the task of change proneness prediction.
There is a significant difference between various models developed using different several set of metrics

All 10 metrics as a feature set yields better result compared to other approaches

Despite different syntax and language semantics of domain specific languages like ST in comparison to that of general purpose languages, classical source code metrics are a good indicator of change proneness
References I

Janez Demšar.  
Statistical comparisons of classifiers over multiple data sets.  

Farideh Fazayeli, Lipo Wang, and Jacek Mandziuk.  
Feature selection based on the rough set theory and expectation-maximization clustering algorithm.  

Karl-Heinz John and Michael Tiegelkamp.  
IEC 61131-3: programming industrial automation systems: concepts and programming languages, requirements for programming systems, decision-making aids.  
References II

A Güneş Koru and Hongfang Liu.
Identifying and characterizing change-prone classes in two large-scale open-source products.

Lov Kumar, Raoul Jetley, and Ashish Sureka.
Source code metrics for programmable logic controller (plc) ladder diagram (ld) visual programming language.

Hongmin Lu, Yuming Zhou, Baowen Xu, Hareton Leung, and Lin Chen.
The ability of object-oriented metrics to predict change-proneness: a meta-analysis.
References III

FJ Malian, JLCM Barbancho, C Leon, A Malian, and A Gomez.
Using industrial standards on plc programming learning.

A. Nair.
Product metrics for iec 61131-3 languages.

Andreas Otto and Klas Hellmann.
Iec 61131: A general overview and emerging trends.

Herbert Prähöfer, Florian Angerer, Rudolf Ramler, Hermann Lacheiner, and Friedrich Grillenberger.
Opportunities and challenges of static code analysis of iec 61131-3 programs.
References IV

Daniele Romano and Martin Pinzger.
Using source code metrics to predict change-prone java interfaces.

Nieke Roos.
Programming plcs using structured text.