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

Lov Kumar[1] Ashish Sureka[2]

[1]NIT Rourkela, India
Email: lovkumar505@gmail.com

[2]ABB India, India
Email: ashish.sureka@in.abb.com

Table of Contents

Table of Contents

Structured Text (ST) - Text-Based Language

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].

Structured Text (ST) - Text-Based Language

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

Source Code Metrics - Change-Proneness

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)

[4] To examine the effectiveness of PCA based and rough set analysis based feature selection techniques

Table of Contents

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]

Code Analysis for PLC Languages

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]

Code Metrics to Predict Change-Proneness

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

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 rought 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

	Version 1		**Version 2**		**Common**	
	CT	**POU**	**CT**	**POU**	**CT**	**POU**
Project 1	214	82	240	82	158	56
Project 2	293	104	344	104	209	71

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

Program Organization Unit (POU) are software units within an application

Experimental Dataset used in Our Study

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]

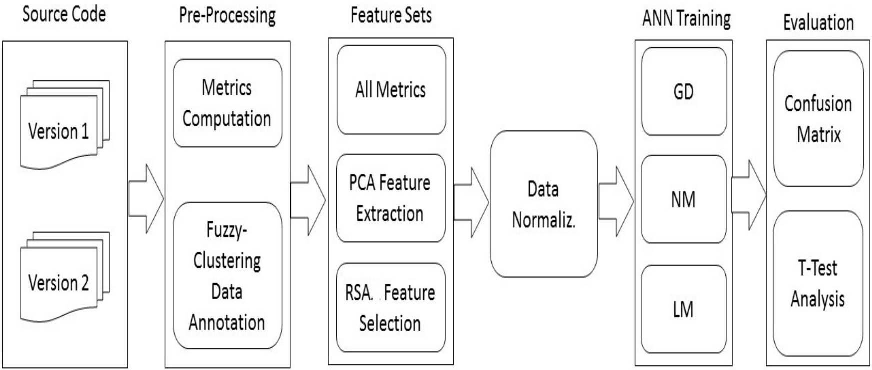
We take two versions of two real world projects in our organization

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

Framework of Proposed Approach



Research Framework and Methodology

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)

Research Framework and Methodology

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

10 source code metrics

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

%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.

10 source code metrics

Calculated Program Length (Cproglen)

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

$$Cproglen = \eta_1 log \eta_1 + \eta_2 log \eta_2 \tag{1}$$

where η_1 and η_1 represents the total number of distinct operators and operands respectively

10 source code metrics

Volume

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

$$Volume = (N_1 + N_2) log(\eta_1 + \eta_2) \quad (2)$$

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

Difficulty

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

$$Difficulty = \frac{\eta_1}{2} * \frac{N_2}{\eta_2} \quad (3)$$

10 source code metrics

Effort

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:

$$CC_{POU} = (\eta_1 + \eta_2) * (\sum_{i=1}^{n} CW_{CT_i} + \sum_{j=1}^{m} CW_{AT_j} + \sum_{k=1}^{o} CW_{IH_k}) \quad (4)$$

where η_1 and η_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:

$$CW_{CT} = \sum_{i=1}^{P} W_i \tag{5}$$

where P represents the number of basic control structures and W_i represents the cognitive weight of i^{th} basic control structure

10 source code metrics

Attribute Cognitive Weight

Attribute Cognitive Weight (CW_{AT}) is used to calculate the complexity of attributes used in the POU. It is calculated using the following equation:

$$CW_{AT} = NPDT * W_{PDT} + NUDT * W_{UDT} \tag{6}$$

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:

$$CW_{IH} = DIT * NOA * CL \tag{7}$$

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:

$$TC_{POU} = \sum_{i=1}^{n} TC_{CT_i} - n \tag{8}$$

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

Table of Contents

Project 1 - Descriptive Statistics (CodeTab)

Metrics	Code Tab Level				
	Min.	Max.	Mean	Median	Std Dev.
Size	1	198	16	6	27.46
ProgLen	4	2752	195.18	83	327.74
Vocabulary	4	227	37.27	30	30.47
Cproglen	2.77	1163.79	126.03	82.32	148.12
Volume	5.55	14929.46	812.74	290.03	1626.02
Difficulty	1	69.59	12.17	8.42	12.52
Effort	5.55	941205.24	24044.21	2402.74	88355.21
CC	4	66057	1804.26	256	6577.14
TC	1	42	3.49	2	5.31
% Comment	0	0.55	0.03	0.01	0.06

Project 2 - Descriptive Statistics (CodeTab)

Metrics	Code Tab Level				
	Min.	Max.	Mean	Median	Std Dev.
Size	1	216	23.29	10	35.18
ProgLen	4	3791	350	114	571.7
Vocabulary	4	245	47.66	34	42.96
Cproglen	2.77	1310.45	178.77	100.88	221.86
Volume	5.55	16265.13	1542.44	395.09	2744.13
Difficulty	1	186.55	17.38	9.39	25.39
Effort	5.55	3034260.33	75949.77	3634.67	284794.55
CC	4	104976	4287.78	378	12755.03
TC	1	42	3.56	2	5.21
% Comment	0	0.55	0.04	0.02	0.06

Code Metrics - Descriptive Statistics and Correlation

The descriptive statistics describes and characterizes the features of the two versions of the two projects

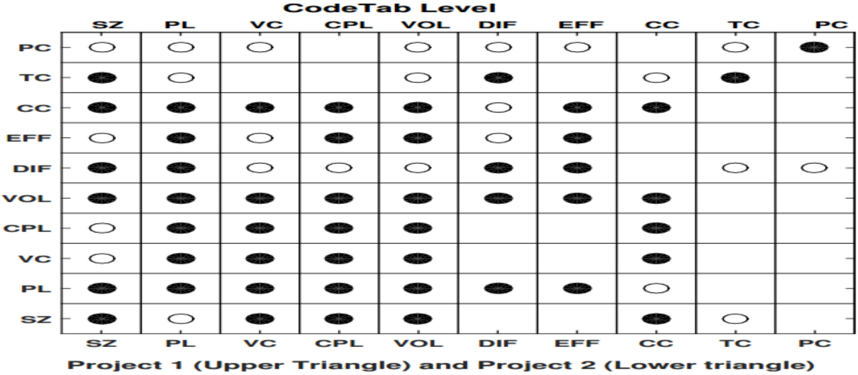
We observe enough variance and dispersion in majority of the variable values

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

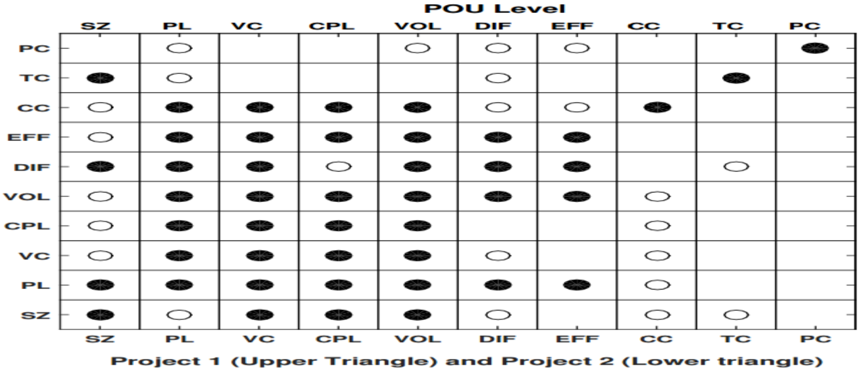
The variance in code complexity metrics shows that some modules or units are more complex than others

Table of Contents

Project 1 (Upper Triangle) and Project 2 (Lower triangle)



Project 1and Project 2 - POU Lavel



Correlations Between 10 Source Code Metrics

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

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

	Project 1		Project 2	
Proneness	**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

Descriptive Statistics of Code Tabs and POUs

	Project 1				**Project 2**			
CP	**Code Tab Level**		**POU Level**		**CodeTab Level**		**POU Level**	
	No.	%	No.	%	No.	%	No.	%
Low (L)	127	80.89	48	84.21	166	79.43	59	81.94
Medium (M)	21	13.38	7	12.28	33	15.79	11	15.28
High (H)	9	5.73	2	3.51	10	4.78	2	2.78

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)

	Project 1							
	CodeTab Level				POU Level			
	PC1	PC2	PC3	PC4	PC1	PC2	PC3	PC4
Size	0.33	0.30	0.32	0.05	0.31	0.43	0.21	0.11
ProgLen	0.36	-0.13	0.02	-0.09	0.36	-0.10	0.04	-0.32
vocabulary	0.32	-0.30	-0.01	0.55	0.34	-0.26	0.19	0.21
Cproglen	0.32	-0.35	0.01	0.43	0.34	-0.30	0.18	0.17
Volume	0.36	-0.19	0.00	-0.17	0.36	-0.15	0.02	-0.32
Difficulty	0.31	0.35	-0.18	0.14	0.34	0.24	-0.15	-0.16
Effort	0.34	-0.12	-0.12	-0.55	0.33	-0.11	-0.24	-0.45
CC	0.33	-0.16	0.22	-0.38	0.31	-0.21	0.13	0.60
TC	0.22	0.62	0.40	0.12	0.21	0.70	0.21	0.09
% cmt	0.23	0.32	-0.80	0.02	0.19	0.10	-0.86	0.35
Eigenvalues	6.98	1.36	0.68	0.49	6.97	1.28	0.89	0.43
% Variance	69.81	13.56	6.76	4.89	69.75	12.83	8.87	4.34
Cumulative % variance	69.81	83.37	90.13	95.02	69.75	82.58	91.45	95.79

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

Rough Set Theory and Expectation Maximization

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

Project	**Level**	**Source Code Metrics**
Project 1	CodeTab	Size, ProgLen, Vocabulary, Cproglen, Difficulty, Effort, TC, %cmt
	POU	Cproglen, Volume, Effort, TC, % cmt,
Project 2	CodeTab	Size, ProgLen, Vocabulary, Cproglen, Volume, Difficulty, TC,% cmt
	POU	Size, Vocabulary, Volume, Difficulty, Effort, CC

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

Table of Contents

Results Based on Accuracy and F-Measure

			Accuracy (%)			F-Measure		
Project	**Level**	**Classifier**	**AM**	**PCA**	*RSA*	**AM**	**PCA**	**RSA**
Project 1	Code Tab	GD	81.53	80.89	79.62	0.93	0.93	0.92
		NM	82.80	79.62	80.89	0.94	0.92	0.93
		LM	84.71	82.17	84.08	0.94	0.93	0.94
	POU	GD	91.23	85.96	85.96	0.97	0.95	0.95
		NM	89.47	85.96	89.47	0.96	0.95	0.96
		LM	92.98	87.72	91.23	0.98	0.96	0.97
Project 2	Code Tab	GD	80.38	78.47	80.38	0.92	0.92	0.92
		NM	85.17	77.51	85.17	0.95	0.91	0.95
		LM	89.00	77.99	89.95	0.96	0.91	0.96
	POU	GD	90.28	83.33	91.67	0.97	0.94	0.97
		NM	93.06	88.89	91.67	0.98	0.96	0.97
		LM	95.83	90.28	93.06	0.99	0.97	0.98

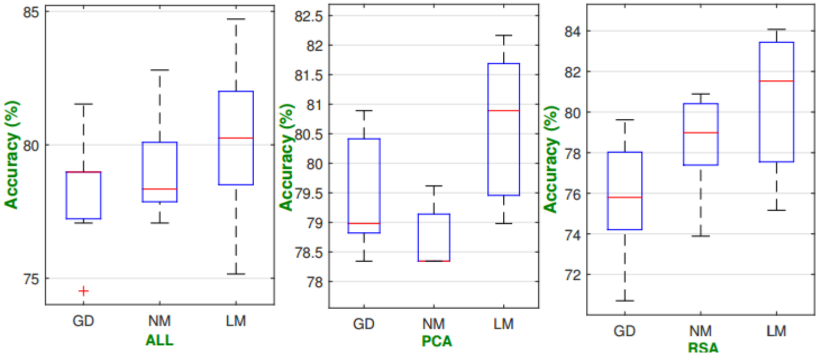
Experimental Results

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

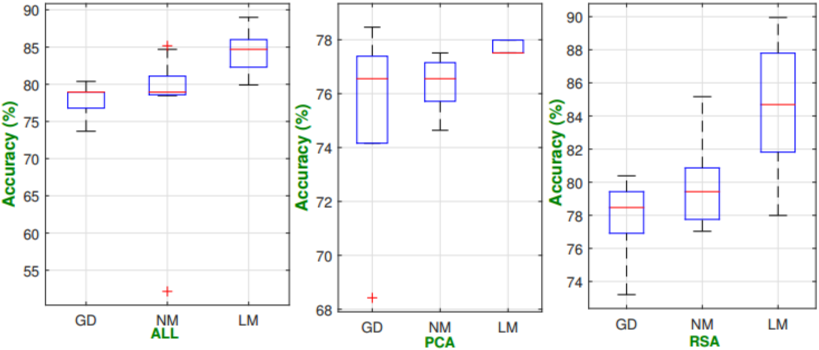
The performance of each prediction model is evaluated in terms of two different performance parameters i.e., accuracy and F-measure

We observe that the classifier is confusing between the classes M and L indicating areas of improvement

Project 1 (Code Tab Level)



Project 2 (Code Tab Level)



Box Plots Analysis

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

t-test - Feature Selection Techniques

Accuracy						
	P-value			**Mean Difference**		
	ALL	PCA	RSA	ALL	PCA	RSA
ALL	NaN	0.00	0.06	0.00	4.80	1.11
PCA	0.00	NaN	0.01	-4.80	0.00	-3.70
RSA	0.06	0.01	NaN	-1.11	3.70	0.00
F-Measure						
	P-value			**Mean Difference**		
ALL	NaN	0.00	0.05	0.00	0.02	0.00
PCA	0.00	NaN	0.007014693	-0.02	0.00	-0.02
RSA	0.05	0.01	NaN	0.00	0.02	0.00

Pairwise t-test

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						
	P-value			Mean Difference		
	GD	NM	LM	GD	NM	LM
GD	NaN	0.05	0.00	0.00	-1.66	-4.11
NM	0.05	NaN	0.00	1.66	0.00	-2.44
LM	0.00	0.00	NaN	4.11	2.44	0.00
F-Measure						
	P-value			Mean Difference		
GD	NaN	0.04	0.00	0	-0.00	-0.01
NM	0.04	NaN	0.00	0.00	0	-0.00
LM	0.00	0.00	NaN	0.016	0.00	0

Pairwise t-test

For each prediction technique a total number of two sets (one for each performance) are used, each with 12 data point [(2 feature selection method + 1 considering all features) * 4 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

Conclusion and Takeaways

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.
Journal of Machine learning research, 7(Jan):1–30, 2006.

Farideh Fazayeli, Lipo Wang, and Jacek Mandziuk.
Feature selection based on the rough set theory and expectation-maximization clustering algorithm.
Rough Sets and Current Trends in Computing RSCTC, pages 272–282, 2008.

Karl-Heinz John and Michael Tiegelkamp.
IEC 61131-3: programming industrial automation systems: concepts and programming languages, requirements for programming systems, decision-making aids.
Springer Science & Business Media, 2010.

References II

A Güneş Koru and Hongfang Liu.
Identifying and characterizing change-prone classes in two large-scale open-source products.
Journal of Systems and Software, 80(1):63–73, 2007.

Lov Kumar, Raoul Jetley, and Ashish Sureka.
Source code metrics for programmable logic controller (plc) ladder diagram (ld) visual programming language.
In *Workshop on Emerging Trends in Software Metrics*, WETSoM, pages 15–21. ACM, 2016.

Hongmin Lu, Yuming Zhou, Baowen Xu, Hareton Leung, and Lin Chen.
The ability of object-oriented metrics to predict change-proneness: a meta-analysis.
Empirical Software Engineering, 17(3):200–242, 2012.

References III

FJ Malian, JLCM Barbancho, C Leon, A Malian, and A Gomez.
Using industrial standards on plc programming learning.
In *Control & Automation, 2007. MED'07. Mediterranean Conference on*, pages 1–6. IEEE, 2007.

A. Nair.
Product metrics for iec 61131-3 languages.
In *Conference on Emerging Technologies Factory Automation (ETFA)*, pages 1–8, Sept 2012.

Andreas Otto and Klas Hellmann.
Iec 61131: A general overview and emerging trends.
Industrial Electronics Magazine, IEEE, 3(4):27–31, 2009.

Herbert Prähofer, Florian Angerer, Rudolf Ramler, Hermann Lacheiner, and Friedrich Grillenberger.
Opportunities and challenges of static code analysis of iec 61131-3 programs.
In *ETFA*, pages 1–8. IEEE, 2012.

References IV

Daniele Romano and Martin Pinzger.
Using source code metrics to predict change-prone java interfaces.
In *Conference on Software Maintenance (ICSM)*, pages 303–312. IEEE, 2011.

Nieke Roos.
Programming plcs using structured text.
In *International Multiconference on Computer Science and Information Technology*, pages 20–22. Citeseer, 2008.