Source Code Metrics for Programmable Logic Controller (PLC) Ladder Diagram (LD) Visual Programming Language

Lov Kumar, Raoul Jetley and Ashish Sureka

ABB Corporate Research, India

lov.kumar@in.abb.com, raoul.jetley@in.abb.com, ashish.sureka@in.abb.com

May 15, 2015
# Table of Contents

1. Research Motivation and Aim
   - Programmable Logic Controllers (PLC)
   - IEC 61131-3
   - Research Motivation
   - Research Aim

2. Related Work and Research Contributions
   - Novel Research Contributions:

3. Proposed Metrics
   - Vocabulary (VC)
   - Size (SZ)
   - Cognitive Complexity (CC)
   - Testing Complexity (TC)

4. Weyuker’s Properties Analysis

5. Comparison
   - Size Vs. Cognitive Complexity
   - Vocabulary Vs. Cognitive Complexity
   - Size Vs. Testing Complexity
   - Vocabulary Vs. Testing Complexity
   - Cognitive Complexity Vs. Testing Complexity

6. Conclusion

7. References
Table of Contents

1. Research Motivation and Aim
   - Programmable Logic Controllers (PLC)
   - IEC 61131-3
   - Research Motivation
   - Research Aim

2. Related Work and Research Contributions
   - Novel Research Contributions:

3. Proposed Metrics
   - Vocabulary (VC)
   - Size (SZ)
   - Cognitive Complexity (CC)
   - Testing Complexity (TC)

4. Weyuker’s Properties Analysis

5. Comparison
   - Size Vs. Cognitive Complexity
   - Vocabulary Vs. Cognitive Complexity
   - Size Vs. Testing Complexity
   - Vocabulary Vs. Testing Complexity
   - Cognitive Complexity Vs. Testing Complexity

6. Conclusion

7. References
Programmable Logic Controllers (PLC)

- Programmable Logic Controller (PLC) are control systems used for factory and industrial automation of electromechanical processes.

- PLCs are programmed using domain specific languages.

- Domain specific languages have several features, programming constructs and notations which are different than general purpose languages.

- International Electro-technical Committee (IEC) developed a standard called as IEC 61131-3 which defines the basic programming elements, syntactic and semantic rules for text-based and graphical or visual programming languages for programming PLCs.
IEC 61131-3 defines graphical languages like Ladder diagram (LD) and Function block diagram (FBD) and text-based languages like Structured text (ST) and Instruction list (IL).

A Domain-Specific Language (DSL) like a Ladder Diagram (LD) is specialized to a particular application domain.

It has specialized features as well as programming constructs not present in general purpose programming languages.
Research Motivation

(a) water and wastewater industry

(b) AC500

Figure 1: Application of Programmable Logic Controller
PLCs for the water and wastewater industry efficiently control Flow, level, pressure\(^1\).

AC500 platform offers different performance levels and is the ideal choice for high availability, extreme environments or safety solutions\(^2\).

\(^1\)http://new.abb.com/drives/segments/water-and-wastewater
\(^2\)http://new.abb.com/plc/programmable-logic-controllers-plcs
The work presented in this paper is motivated by the need to define practical and rigorous metrics for PLC domain specific languages.

The specific aim of our study is to define size, vocabulary, cognitive complexity and testing complexity of Ladder Diagram (LD).
# Table of Contents

1. Research Motivation and Aim
   - Programmable Logic Controllers (PLC)
   - IEC 61131-3
   - Research Motivation
   - Research Aim

2. Related Work and Research Contributions
   - Novel Research Contributions:

3. Proposed Metrics
   - Vocabulary (VC)
   - Size (SZ)
   - Cognitive Complexity (CC)
   - Testing Complexity (TC)

4. Weyuker’s Properties Analysis

5. Comparison
   - Size Vs. Cognitive Complexity
   - Vocabulary Vs. Cognitive Complexity
   - Size Vs. Testing Complexity
   - Vocabulary Vs. Testing Complexity
   - Cognitive Complexity Vs. Testing Complexity

6. Conclusion

7. References
**Related Work and Research Contributions**

- **Lucas et al. (2005)** present methods of measuring the size and complexity of PLC programs in different logic control design methodologies [1].
- **Cardoso et al. (2006)** survey of process complexity metrics in the domain of business process modelling [2].
- **Gharieb et al. (2006)** proposed quality metrics involves the criteria of simplicity, re-configurability, reliability, and flexibility [3].
Gruhn et al. (2006) discusses how existing research results on the complexity of software can be extended in order to analyze the complexity of business process models [4].

Waters et al. (2008) describes a method for creating tools to calculate software metrics for ladder logic [5].

Nair et al. (2012) define a set of product metrics that can be used for managing the software project development using IEC 61131-3 languages [6].
Novel Research Contributions:

- We define **vocabulary**, **size**, **cognitive complexity** and **testing complexity** of Ladder Diagram (LD) programs.

- These all metrics are defined using factors like number of distinct **operators** and total number of **operators** and **operands**, number of **rungs**, **weights** of different basic control structures, **structure of the program and control flow**.

- we present an application of **Weykur’s property** to validate the metrics and evaluate the number of properties satisfied by the proposed metric.
Table of Contents

1. Research Motivation and Aim
   - Programmable Logic Controllers (PLC)
   - IEC 61131-3
   - Research Motivation
   - Research Aim

2. Related Work and Research Contributions
   - Novel Research Contributions:

3. Proposed Metrics
   - Vocabulary (VC)
   - Size (SZ)
   - Cognitive Complexity (CC)
   - Testing Complexity (TC)

4. Weyuker’s Properties Analysis

5. Comparison
   - Size Vs. Cognitive Complexity
   - Vocabulary Vs. Cognitive Complexity
   - Size Vs. Testing Complexity
   - Vocabulary Vs. Testing Complexity
   - Cognitive Complexity Vs. Testing Complexity

6. Conclusion

7. References
Proposed Metrics

- Vocabulary (VC)
- Size (SZ)
- Cognitive Complexity (CC)
- Testing Complexity (TC)
Vocabulary (VC)

\[ \text{Vocabulary}(VC) = \eta_1 + \eta_2 \]  \hspace{1cm} (1)

where

- \( \eta_1 \): Total number of distinct operators.
- \( \eta_2 \): Total number of distinct operands.
- **Operator**: symbol which performs specific operations on one or multiple operands and returns result i.e., input contact, output coil, counter, timer etc.
- **Operand**: symbol used to represent a ladder diagram element.
Figure 2: Sequence Control Structure

- $\eta_1 = 2$ (input contact, output coil)
- $\eta_2 = 4$ ($I_1, I_2, I_3$ and $O_1$)
- Vocabulary (VC) = $\eta_2 + \eta_2 = 2 + 4 = 6$
Size (SZ)

- We define size as the number of executable basic control structures in a Ladder Diagram program.
- The proposed metric for size is similar to the LOC (Lines of Code) metric used in general purpose text-based programming languages which counts the number of executable statements in a program.
- The size of the LD in Figure 2 is 4 because there are 3 inputs and 1 output element.
Cognitive Complexity (CC)

- We define Cognitive Complexity (CC) as a metric to measure how easy or difficult it is to understand.

\[
CC = (N_i + N_o) \times W_c
\]  

- \(N_i\): Total number of distinct input variables.
- \(N_o\): Total number of distinct input output variables.
- \(W_c\): Total cognitive weight of the given ladder diagram.
Cognitive weight

\[ W_c = \sum_{i=1}^{N_r} W_i \]  \hspace{1cm} (3)

- \( W_i \): weight of rung \( i \).
- \( N_r \): Total numbers of number of rungs.
- Table 1 shows the basic control structures and their cognitive weight \( (W_i) \) of PLC ladder diagram.
## Cognitive weight ($W_i$)

### Table 1: Basic Control Structures and their Cognitive Weight

<table>
<thead>
<tr>
<th>Category</th>
<th>BCS</th>
<th>Weight</th>
</tr>
</thead>
<tbody>
<tr>
<td>Sequence</td>
<td>SEQ</td>
<td>1</td>
</tr>
<tr>
<td>Branch</td>
<td>Split</td>
<td>2</td>
</tr>
<tr>
<td></td>
<td>JUMP</td>
<td>2</td>
</tr>
<tr>
<td>Iteration</td>
<td>For</td>
<td>3</td>
</tr>
<tr>
<td></td>
<td>Timer</td>
<td>3</td>
</tr>
<tr>
<td></td>
<td>counter</td>
<td>3</td>
</tr>
<tr>
<td>Embedded Component</td>
<td>Arithmetic Function</td>
<td>2</td>
</tr>
<tr>
<td></td>
<td>Control function</td>
<td>2</td>
</tr>
<tr>
<td></td>
<td>Data comparison</td>
<td>2</td>
</tr>
<tr>
<td></td>
<td>Data movement</td>
<td>2</td>
</tr>
<tr>
<td></td>
<td>Data conversion function</td>
<td>2</td>
</tr>
<tr>
<td></td>
<td>Binary function</td>
<td>2</td>
</tr>
<tr>
<td></td>
<td>Sequerencer function</td>
<td>2</td>
</tr>
<tr>
<td>Concurrency</td>
<td>Time interrupt</td>
<td>4</td>
</tr>
<tr>
<td></td>
<td>Fault interrupt</td>
<td>4</td>
</tr>
<tr>
<td></td>
<td>Input driven Interrupt</td>
<td>4</td>
</tr>
</tbody>
</table>
Cognitive weight ($W_i$)

- Each rungs of the ladder diagram contains $M$ number of basic control structure. Equation 3 can be written as Equation 4:

$$W_c = \sum_{i=1}^{N_r} \sum_{j=1}^{M} W_{i,j} \quad (4)$$

- A basic control structure may consists of $P$ layers of nested basic control structures. Hence, to incorporate nested control structure, we extend Equation 4 to Equation 5:

$$W_c = \sum_{i=1}^{N_r} \left[ \prod_{k=1}^{p} \sum_{j=1}^{M} W_{i,j} \right] \quad (5)$$
Example

(a) Ladder Diagram with Simple Control Structure  
(b) Ladder Diagram with Nested Control Structure

Figure 3: Ladder Diagram with Simple and Nested Control Structures
Example

- Figure 3 shows two Ladder Diagram programs.
- The number and types of basic control structures in both the diagrams are same.
- However, the structure of both the programs are different.
- The Ladder Diagram in Figure 3b is more cognitively complex as compared to the Ladder Diagram of Figure 3a due to the nesting structure and depth.
- The CC value of the ladder diagram in Figure 3a is $CC = (7 + 1) \times W_c = 8 \times 7 = 56$.
- The CC value of the ladder diagram in Figure 3b is $CC = (7 + 1) \times W_c = 8 \times 15 = 120$. 
The Testing Complexity (TC) is a measure of testability (how much difficult to test).

\[ TC = \sum_{i=1}^{N_r} TC_i \]  

- **TC_i**: Testing complexity of rung \( i \).
- **N_r**: Total number of rungs in the ladder diagram.
- A rung which contains more than one control flow path can be classified as a *split* component.
- In case of single split component, at-least one and at-most two control paths are possible. Hence, for every split component, \( 2^2 - 1 \) is added to TC.
Example

In Figure 5, there is one split component ($I_2$ and $I_3$ in parallel). Hence the total number of control flow path in $Q$ is $2^2 - 1 = 3$.

1. $I_1=True$, $I_2=True$, $I_3=False$ $\rightarrow$ $O_1=True$
2. $I_1=True$, $I_2=False$, $I_3=True$ $\rightarrow$ $O_1=True$
3. $I_1=True$, $I_2=True$, $I_3=True$ $\rightarrow$ $O_1=True$
Table of Contents

1. Research Motivation and Aim
   - Programmable Logic Controllers (PLC)
   - IEC 61131-3
   - Research Motivation
   - Research Aim

2. Related Work and Research Contributions
   - Novel Research Contributions:

3. Proposed Metrics
   - Vocabulary (VC)
   - Size (SZ)
   - Cognitive Complexity (CC)
   - Testing Complexity (TC)

4. Weyuker’s Properties Analysis

5. Comparison
   - Size Vs. Cognitive Complexity
   - Vocabulary Vs. Cognitive Complexity
   - Size Vs. Testing Complexity
   - Vocabulary Vs. Testing Complexity
   - Cognitive Complexity Vs. Testing Complexity

6. Conclusion

7. References
Weyuker’s Properties

- Weyuker’s define a formal list of properties for software metrics.
- It is used by several researchers to evaluate their proposed software metrics.
- We present our analysis of Weyuker’s properties on our metrics.
- The objective of analyzing Weyuker’s properties is to validate our proposed metrics.
Property 1:

\( \exists P \)(\( \exists Q \))(CC(P) \neq CC(Q))

Property 1: \( \exists P \)(\( \exists Q \))(CC(P) \neq CC(Q))

![Diagram of PLC Ladder Diagram: (a) Sequential (b) Split]

Figure 5: Examples of PLC Ladder Diagram: (a) Sequential (b) Split

- The CC value of \( P \) and \( Q \) are \( 1 \times (3 + 1) = 4 \) and \( (1 + 2) \times (3 + 1) = 12 \).

- The example demonstrates that there exists two programs which have different CC value.
Property 2: Let $c$ be a non-negative number. Then there are finitely many programs of complexity $c$.

- In PLC ladder diagram, the number of distinct elements are fixed.
- All program uses this fixed number of component to draw their ladder diagram.
- At one instance, the component will get repeated and resulting ladder diagrams have same number of inputs and output element.
- But the possible number of structure of the ladder diagrams are infinite, that give infinite value of $W_c$.
- Hence, We conclude that Property 2 is not satisfied for CC metric.
Property 3: There are distinct programs P and Q such that $(CC(P) = CC(Q)$

![Diagram](image1.png)

Figure 6: Examples of PLC Ladder Diagram (a) Function (b) Split

- The CC value of $P$ and $Q$ are $(1 + 2) \times (3 + 2) = 15$ and $(1 + 2) \times (4 + 1) = 15$.
- CC satisfy Property 3.
Property 4: $(\exists P)(\exists Q) \ (P=Q \ & \ CC(P) \neq CC(Q))$

Figure 7: Examples of PLC Ladder Diagram showing Sequential Structure and Pre-Defined Function Element

- In both programs, the output $O_1$ should be true when input $I_1$ is True and input $I_2$ is true, otherwise output should be false.
- CC value of $P$ and $Q$ are $1 \times (2+1) = 3$ and $(2+1) \times (2+3) = 15$ respectively. Hence, CC satisfies Property 4.
Property 5: 

\( (\forall P)(\forall Q) \ (CC(P) \leq CC(P; Q) \ & \ CC(Q) \leq CC(P; Q)) \)

- We demonstrate the satisfy-ability of Property 5 using concatenation.
- If we concatenate two PLC ladder diagrams, the number of rungs is always greater or equal to number of rungs of largest ladder diagram.
Property 6:

Property 6a: \((\exists P)(\exists Q)(\exists R) (CC(P) = CC(Q) & CC(P; R) \neq CC(Q; R))\)

Property 6b: \((\exists P)(\exists Q)(\exists R) (CC(P) = CC(Q) & CC(R; P) \neq CC(R; Q))\)

- Figure 8 shows five PLC ladder diagrams: \(P\), \(Q\), \(R\), concatenation of \(P\) and \(Q\) \((P; R)\), and concatenation of \(Q\) and \(R\) \((Q; R)\).

- The CC value both ladder diagram \(P\) and \(Q\) are same i.e., \(1 \times (2 + 1) = 3\). The CC values after concatenation of \(R\) with \(P\) and \(Q\) are \(CC(P; R) = (1 + 2) \times (3 + 1) = 12\), and \(CC(Q; R) = (1 + 1)(3 + 2) = 10\). Hence, CC satisfy Property 6.
Property 6:

Figure 8: Examples of PLC Ladder Diagram showing simple structure and concatenation.
Property 7: There are program bodies $P$ and $Q$ such that $Q$ is formed by permuting the order of the statements of $P$, and $(CC(P) \neq CC(Q))$

- The property ensures that potential interaction is evaluated by the measure and is sensitive to the interaction.
- CC satisfies this property because the permutation of control flows can result in different control flows structure (resulting in change in structure), hence making the CC value different.
Property 8: If $P$ is a renaming of $Q$, then ($CC(P) = CC(Q)$)

- CC value of Ladder diagram depends on the control flow structure of the program as well as the cognitive weights of the various elements used in the program.

- The renaming of various elements of the ladder diagram does not change the structure and number and type of distinct operators. Hence, CC satisfies Property 8.
Property 9: \((\exists P)(\exists Q)(CC(P) + CC(Q) < CC(P; Q))\)

- Figure 9 show the three PLC ladder diagrams: \(P\), \(Q\), and concatenation of \(P\) and \(Q\) (\(P; Q\)).
- The CC value both ladder diagram \(P\) and \(Q\) are \(1 \times (2 + 1) = 3\).
- The summation value of CC for \(P\) and \(Q\) ladder diagram is \(6\) which is less than the CC vale of \(P; Q\) \((CC(P; Q) = (1 + 2) \times (3 + 1) = 12\))
- Hence, CC metric satisfies Property 9.
Property 9:

(a) P

(b) Q

(c) P;Q

Figure 9: Examples of two sequential PLC Ladder Diagram and their concatenation.
### Table 2: Weyuker Properties and Source Code Metrics. The symbol (denoting yes or no) in Each Cell shows if the Property is Satisfied or Not

<table>
<thead>
<tr>
<th>Properties</th>
<th>Size</th>
<th>VC</th>
<th>CC</th>
<th>TC</th>
</tr>
</thead>
<tbody>
<tr>
<td>Property 1</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td>Property 2</td>
<td>✓</td>
<td>✓</td>
<td>×</td>
<td>×</td>
</tr>
<tr>
<td>Property 3</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td>Property 4</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td>Property 5</td>
<td>✓</td>
<td>×</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td>Property 6</td>
<td>✓</td>
<td>×</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td>Property 7</td>
<td>×</td>
<td>×</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td>Property 8</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td>Property 9</td>
<td>×</td>
<td>×</td>
<td>×</td>
<td>×</td>
</tr>
</tbody>
</table>
Table of Contents

1 Research Motivation and Aim
   • Programmable Logic Controllers (PLC)
   • IEC 61131-3
   • Research Motivation
   • Research Aim

2 Related Work and Research Contributions
   • Novel Research Contributions:

3 Proposed Metrics
   • Vocabulary (VC)
   • Size (SZ)
   • Cognitive Complexity (CC)
   • Testing Complexity (TC)

4 Weyuker’s Properties Analysis

5 Comparison
   • Size Vs. Cognitive Complexity
   • Vocabulary Vs. Cognitive Complexity
   • Size Vs. Testing Complexity
   • Vocabulary Vs. Testing Complexity
   • Cognitive Complexity Vs. Testing Complexity

6 Conclusion

7 References
The size of the both ladder diagrams displayed in Figure 3 is 16 whereas the Cognitive Complexity of one program is 56 and the other is 120.

we conclude that two programs of similar size can have different cognitive complexity.
Vocabulary Vs. Cognitive Complexity

- The vocabulary for both the ladder diagram programs of Figure 3 is 10 whereas the cognitive complexity values are significantly different.

- We conclude that two programs of similar vocabulary can have different cognitive complexity.
Size Vs. Testing Complexity

- The size of the both ladder diagrams displayed in Figure 5 is 4 whereas the Testing Complexity of one program is 1 and the other is 3.

- we conclude that two programs of similar size can have different Testing Complexity.
Vocabulary Vs. Testing Complexity

▶ The Vocabulary of the both ladder diagrams displayed in Figure 5 is 6 whereas the Testing Complexity of one program is 1 and the other is 3.

▶ we conclude that two programs of similar Vocabulary can have different Testing Complexity.
Cognitive Complexity Vs. Testing Complexity:

- The Cognitive Complexity of the both ladder diagrams displayed in Figure 6 is 15 whereas the Testing Complexity of one program is 1 and the other is 3.
- we conclude that two programs of similar Cognitive Complexity can have different Testing Complexity.
Table of Contents

1 Research Motivation and Aim
   - Programmable Logic Controllers (PLC)
   - IEC 61131-3
   - Research Motivation
   - Research Aim

2 Related Work and Research Contributions
   - Novel Research Contributions:

3 Proposed Metrics
   - Vocabulary (VC)
   - Size (SZ)
   - Cognitive Complexity (CC)
   - Testing Complexity (TC)

4 Weyuker’s Properties Analysis

5 Comparison
   - Size Vs. Cognitive Complexity
   - Vocabulary Vs. Cognitive Complexity
   - Size Vs. Testing Complexity
   - Vocabulary Vs. Testing Complexity
   - Cognitive Complexity Vs. Testing Complexity

6 Conclusion

7 References
Conclusion

▶ We propose vocabulary, size, cognitive complexity and testing complexity metrics for a visual programmable control logic programming language.

▶ The proposed metric is based on various factors such as the number of distinct operators and operands, the total number of operators and operands, cognitive weights of various control structures and the overall control flow of the program.

▶ We apply Weyuker’s properties on the proposed metrics and present the list of properties which are satisfied and the ones which are not satisfied.

▶ Cognitive and testing complexity of programs can vary significantly depending on the structure and the basic elements used.
## Table of Contents

1. **Research Motivation and Aim**  
   - Programmable Logic Controllers (PLC)  
   - IEC 61131-3  
   - Research Motivation  
   - Research Aim

2. **Related Work and Research Contributions**  
   - Novel Research Contributions:

3. **Proposed Metrics**  
   - Vocabulary (VC)  
   - Size (SZ)  
   - Cognitive Complexity (CC)  
   - Testing Complexity (TC)

4. **Weyuker’s Properties Analysis**

5. **Comparison**  
   - Size Vs. Cognitive Complexity  
   - Vocabulary Vs. Cognitive Complexity  
   - Size Vs. Testing Complexity  
   - Vocabulary Vs. Testing Complexity  
   - Cognitive Complexity Vs. Testing Complexity

6. **Conclusion**

7. **References**
References


References II

Matthew Waters, Ken Young, and Ira D. Baxter.
Calculating software metrics for ladder logic.
In Joaquim Filipe, Juan Andrade-Cetto, and Jean-Louis Ferrier, editors, ICINCO-RA (2), pages 143–150, 2008.

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

Sanjay Misra and Ibrahim Akman.
Applicability of weyuker’s properties on oo metrics: Some misunderstandings.

E.J. Weyuker.
Evaluating software complexity measures.
References III

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

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

Norman Fenton and James Bieman.  
*Software metrics: a rigorous and practical approach.*  
CRC Press, 2014.

Michele Lanza and Radu Marinescu.  
*Object-oriented metrics in practice: using software metrics to characterize, evaluate, and improve the design of object-oriented systems.*  
Questions?