Software Improvement Group

# SIG/TÜVIT EVALUATION CRITERIA TRUSTED PRODUCT MAINTAINABILITY FOR PLC TECHNOLOGIES

**ADDENDUM | Version 2.0**

SOFTWAREIMPROVEMENTGROUP.COM

# TABLE OF CONTENTS

# 1. INTRODUCTION

This document is a companion to document SIG/TÜViT Evaluation Criteria Trusted Product Maintainability. The SIG/TÜViT Evaluation Criteria for the quality mark TÜViT Trusted Product Maintainability are intended for the standardized evaluation and certification of the technical quality of the source code of software products.

This guidance document provides explanation to software producers about the measurement method of SIG applies for evaluation in the context of ***Programmable Logic Controllers (PLC)***. The goal of this document is to explain the relationship between the concepts from the IEC 61131-3 for Programmable Logic Controllers to the concepts from the SIG Maintainability Model. Also, this document explains how the measurements used are performed on PLC code.

# 2. DEFINITIONS

## 2.1 SYSTEM

A typical industrial automation setup consists of one or multiple PLCs that all perform tasks to support a process in the physical world. A PLC system can be defined as the set of source code artefacts that run on the PLCs for which the system owner seeks evaluation.

## 2.2 COMPONENT

A component is defined by the SIG Maintainability Model as a subdivision of a system in which source code modules are grouped together based on a common trait. In a PLC system, source code is typically divided over a set of PLCs where each PLC is responsible for a certain task in the physical world. Also, PLC code can be grouped in libraries that can be reused in multiple PLC's. Additionally, naming conventions and folder structures can be used to distinguish source code artifacts based on architecture concepts. An example of this is a PLC system where source code artifacts are named after concepts from the ISA-88 reference architecture like Units, Equipment Module, Control Module, etc.

Typically, multiple perspectives can be taken on the component view of a software system. In the context of analyzing components using the SIG/TÜViT Evaluation Criteria, the development view is leading in determining how source code is structured into components.

## 2.3 MODULE

The concept of a module in the SIG Maintainability can be referred to as a file containing PLC code. The IEC 61131-3 refers to this concept as a Program Organisation Unit (POU). Examples of constructs from IEC 61131-3 that are typically defined in a single file are Programs, Function Blocks, Functions and User-Defined Types. All constructs/POU that are defined in the same file are considered part of the same module in the SIG Maintainability Model.

The following table lists examples of vendor-specific Program Organisation Units that are counted as modules in the SIG Maintainability Model:

| Vendor | POU |
|---|---|
| ABB | Control Module, Function Block, Data Type |
| Rockwell Automation | Function Block, Routine, Tag, Add-on Instruction |
| Siemens | User-defined Type, Function, Function Block |
| Schneider Electric | Function Block, Function, Subroutine |

## 2.4 UNIT

The notion of unit in the SIG Maintainability Model is defined as the smallest named piece of executable code. The IEC 61131-3 does not define a common mechanism to group and name pieces of source code within a Program Organisation Unit. Typically, vendors of distributed control systems have introduced their own mechanism to allow developers to group and name source code within a Program Organisation Unit. These mechanisms assist software engineers with analysing and navigating the source code of a POU. For evaluation of source code against the SIG Maintainability Model, the mechanisms that are provided by the vendor of the used Distributed Control System are used for performing measurements at the unit level.

The following table lists examples of vendor-specific code grouping mechanisms that are counted as units in the SIG Maintainability Model:

| Vendor | Code grouping mechanism within a POU |
|---|---|
| ABB | Code Block |
| Rockwell Automation | Subroutine, Sheet |
| Siemens | Region, Network |
| Schneider Electric | Section |

# 3. GUIDANCE FOR PRODUCERS

For measuring software metrics on PLC source code artefacts SIG first converts the vendor specific format to the generic textual format as described in IEC 61131-3. For example, input and output configuration of a function block is converted to VAR, VAR_IN and VAR_OUT declarations. Where applicable, SIG converts the vendor-specific format to comply with the Structured Text language specification.

## 3.1 VOLUME

All source lines containing statements that comply with the IEC 61131-3 are counted for the volume of the system. Line comments, block comments and blank lines are not counted.

| Language | Volume determination method |
|---|---|
| Structured Text / Instruction List | Line of code |
| Ladder Logic | A rung is counted as the equivalent of a line of code |
| Functional Block Diagram | A function block is counted as the equivalent of a line of code |
| Sequential Function Chart | A step is counted as the equivalent of a line of code |

To compare a system's volume to the volumes of systems written in different technologies, a normalisation is performed based on industry-average productivity in these technologies. For PLC languages, the value of 228,000 Lines of Code in 20 person years is being used.

## 3.2 DUPLICATION

No difference with duplication measurements in other technologies.

## 3.3 UNIT SIZE

The contents of the following language constructs are evaluated for the unit size metric:

- ACTION,
- CODEBLOCK,
- FUNCTION / FUNCTION_BLOCK / NETWORK,
- REGION,
- ROUTINE,
- STRUCT,
- various VAR structures (such as VAR_GLOBAL, VAR_EXTERNAL, VAR_TEMP).

The contents of constructs VAR_IN, VAR_OUT and VAR_IN_OUT are explicitly ignored for the unit size metric because they describe the interface of the module rather than the implementation of the module. For language constructs that allow nesting (e.g. REGION), the most outer construct is counted as a unit.

## 3.4 UNIT COMPLEXITY

For the unit complexity metric, the same definition of a unit applies as for the unit size metric.

The following language constructs are counted as branch points:

| Construct | Tokens |
|---|---|
| Binary operators | AND, OR, XOR |
| Conditional statements | IF, ELSIF, XIC, XIO, JMP |
| Loop statements | REPEAT, WHILE, FOR |
| Select statements | CASE (except the default case) |

## 3.5 UNIT INTERFACING

In PLC languages, the interface of a unit is defined at the level of the POU. The interface of the POU applies to all units in that POU. The only exception is when the SUBROUTINE construct in Ladder Logic is used. A subroutine allows for defining its own interface with parameters within a POU.

## 3.6 MODULE COUPLING

No difference with module coupling measurements in other technologies.

## 3.7 COMPONENT BALANCE

No difference with component balance measurements in other technologies.

## 3.8 COMPONENT INDEPENDENCE

No difference with component independence measurements in other technologies.

Software Improvement Group

Fred. Roeskestraat 115
1076 EE Amsterdam
The Netherlands

www.softwareimprovementgroup.com
marketing@softwareimprovementgroup.com

Getting software right for a healthier digital world