System Requirements Specification

(TINF19C, SWE I, 2020 / 2021)

Project: PLCopen-Editor

Customer: Rentschler & Holder
Rotebühlplatz 41
70178 Stuttgart

Supplier: Team 1 (Franziska Kopp, Leonie de Santis, Mouaz Tabboush, Elian Yildirim)

Version	Date	Author	Comment
0.1	30.10.2020	Elian Yildirim	Created
0.2	01.11.2020	Elian Yildirim	Added Introduction
0.3	02.11.2020	Elian Yildirim	Added functional Requirements
0.4	03.11.2020	Elian Yildirim	Added Non-functional Requirements
0.5	03.11.2020	Elian Yildirim	Added Product Data
0.6	03.11.2020	Elian Yildirim	Added System Environment
1.1	07.11.2020	Elian Yildirim	Error correction with the help of Mr. Rentschler's feedback

Contents

1. Introduction

The PLCopen-Editor is an application for the web and is used to create programs for the Open-PLC-Runtime with a graphical user interface. The programs shall be compiled by different graphical components, based on the IEC 61131-3 standard, and shall support the two programming languages FBD (Function Block Diagram) and SFC (Sequential Function Chart). Furthermore, the PLCopen-Editor could support the languages LD (Ladder Logic), IL (Instruction List) and ST (Structured Text), but the main languages, the editor shall support, are FBD and SFC.
PLCopen XML (Extensible Markup Language, Version 7.0.3) or AML (Automation Markup Language, Version CAEX130) files shall be exported after the creation of a program. These formats shall compile the assembled components on the user interface to an executable and compatible program. PLCopen XML and AML files also shall be imported into the editor and displayed graphically.

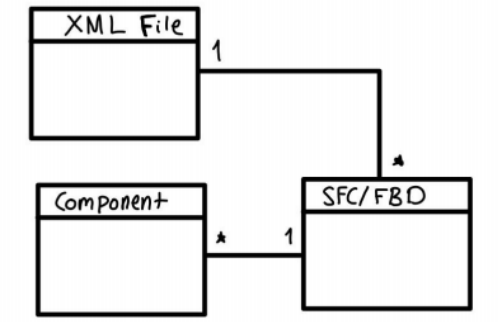

figure 1: The PLCopen-editor (?)

1.1 Product environment

The PLCopen-Editor is used in automation technology for the control and regulation of processes. The programming languages LAD, FBD, AQL, ST and SFC should be supported. The focus, however, should be on the languages FBD and SFC, since these are the graphical representation for describing sequential processes.
The PLCopen-Editor can also display a program in the PLCopen XML [2] language and thus both export and import corresponding PLCopen XML. In addition, a created program can also be exported and imported as AML [3] file.
These both formats (PLCopen XML and AML) are used for the exchange between development environments. AML enables the XML-based representation of automation systems and objects through a transfer of technical data. The PLCopen XML format contains the logical description and information about the different PLC programming languages and the program organization units (POU) created by the user within graphical information (like position and connection), functions, function blocks etc.
SFC is a graphical description of sequential processes and is used to break a process down into clearly arranged parts and to describe the control between these parts. The steps can be used independently in another IEC 61131-3 defined language.
FBD is another graphical programming language for PLC and works with a list of networks. Each network has a structure that can contain logical expressions, function block calls, jumps or return instructions.

1.2 Use Cases

The users of the PLCopen-Editor shall use various functions and areas of the editor, which are summarized in the following graphic:

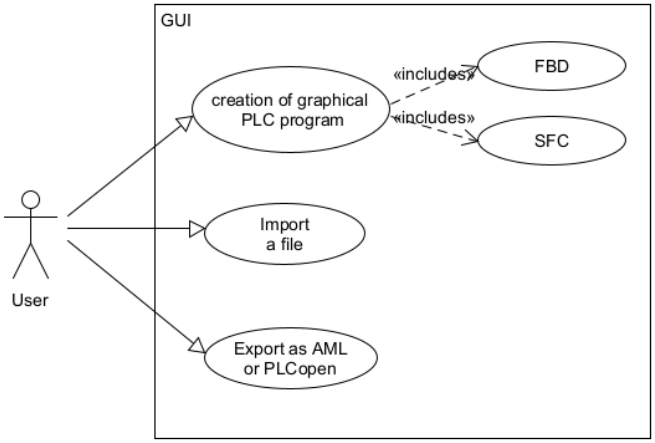

figure 2: Use case overview diagram

1.2.1 <UC.001>: Create PLC program

Related Business Process:	<BP.001>: PLC programming
Use Cases Objective:	User wants to create a graphical PLC program. The User can choose between the programming languages FBD and SFC
System Boundary:	Program itself is the system boundary.
Precondition:	The GUI has to run without errors.
Postcondition on success:	If an error occurs, an easily understandable error message should be communicated.
Involved Roles:	User, Editor
Triggering Event:	User opens the GUI

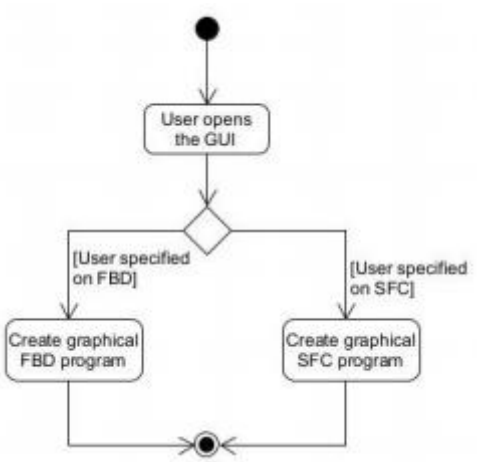

figure 3: <UC.001> Create PLCopen program

1.2.2 <UC.002>: Import file

Related Business Process:	<BP.002>: File conversion
Use Cases Objective:	User wants to import an OpenPLC-XML or AutomationML file to load the PLC-program
System Boundary:	Program itself is the system boundary.
Precondition:	Imported file must be valid and without errors
Postcondition on success:	The PLC program was loaded completely.
Involved Roles:	User, PLCopen/AML file, Importer
Triggering Event:	User opens the GU and want to import a file

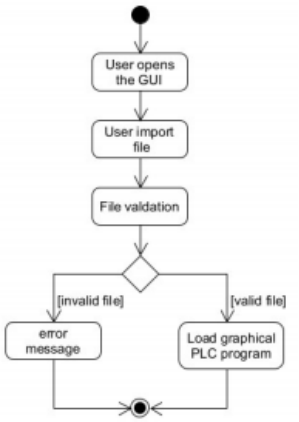

figure 4: <UC.002> Import file

1.2.3 <UC.003>: Export as PLCopen or AML

Related Business Process:	<BP.003>: Data storage
Use Cases Objective:	User wants to export the PLC program as PLCopen XML or AutomationML to exchange their program between development environments
System Boundary:	Program itself is the system boundary
Precondition:	Created PLC program must be without errors
Postcondition on success:	The browser shall not be closed until the export is completed
Involved Roles:	PLC program, Exporter
Triggering Event:	User finisched the graphical PLC program and decide to save in PLCopen XML or AutomationML

figure 5: <UC.003> Export as PLCopen or AML

2. Product Requirements

The following functionalities shall be supported by the PLCopen-Editor.

2.1 /LF10/GUI Components

The GUI shall have a homepage where the user can create or load a new project. Furthermore there shall be a page for the project overview, where the user can add new sections to the project. These sections shall then be edited in the editor and "filled" with function blocks etc.

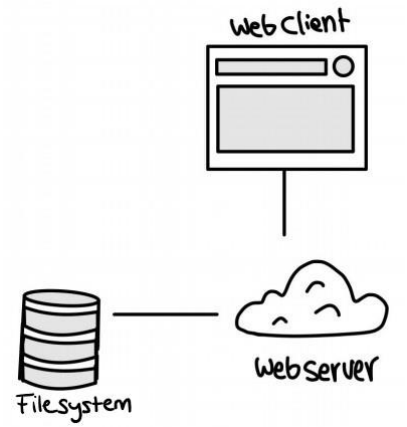

figure 6: Logical component diagram

2.2 /LF20/Connectors

The user interface shall offer through the integration of connectors a simple and clear editing and creation of the programs with the function blocks. Therefor, the function blocks should have input and output pins. These shall be connected to each other. Also, the user should be able to create input and output variables for a function block in form of a table.

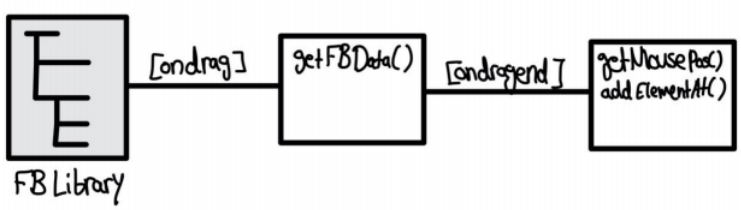

figure 7: Drag and drop / connectors diagram

2.3 /LF30/Error display

The user should be informed via the user interface whether there is an error or a warning.

2.3.1 Errors

Errors shall popup in form of an alert, if the user wants to do something invalid, that leads to a conflict with the backend or bad logic that cannot be exported. The user should be able to print errors in form of a logfile.

2.3.2 Warnings

Warnings shall be marked in yellow with an exclamation mark, if the user could do something better. Warnings are not necessary for an export and are accepted. They are no real errors.

2.4 /LF40/File format analysis

The user shall be informed via the user interface if he wants to import an incompatible XML file into the project.

2.5 /LF50/File validation

The File validation shall check if the imported file is convertible and corresponds to the PLCopen XML schema or AML. In case of violation, an error message is displayed and the user is asked to import a new file. If the file is valid, it can be imported.

2.6 /LF60/Languages support

For the time being, the GUI should be designed for the programming languages SFC and FBD. The user should be able to design, import and export programs in these two languages.

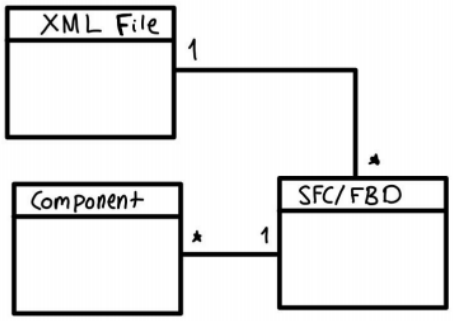

figure 8: Static data structure diagram

2.7 /LF70/Import PLCxml and AML

The importer should create a graphical PLC program from the imported file. The imported format of the file has to be PLCxml or AML. Otherwise, there would occur an error message. If the file is imported completely, the user can edit the PLC program.

2.8 /LF80/Export

The exporter should save the created PLC program as PLCopen XML or AML file. If an error occurs, a corresponding error message should be displayed. If the export was completed successfully, the web application can be closed or continued. The exported file can be imported into the PLCopen editor or the AutomationML editor.

2.9 /LF90/Function blocks

Function blocks form the basis of the processing in the PLC-editor and the logic of a PLC program and have input and output pins for their connection, so that they can build the program and its logic. There are many function blocks like for example SR [4] or RS [4] etc. Therefor, the existing library can be used.

figure 9: examples of function blocks

2.9.1 SR function block

This function block maintains its output in one of two stable states TRUE or FALSE. The output can be set or reset by applying a TRUE signal to the Set or Reset inputs. If both inputs are TRUE the output is TRUE.[4]

2.9.2 RS function block

This function block maintains its output in one of two stable states TRUE or FALSE. The output can be set or reset by applying a TRUE signal to the Set or Reset inputs. If both inputs are TRUE the output is FALSE.[4]

2.10 /LF100/Navigation

The user shall navigate through his project with the help of a navigation bar. There, the user can see the project overview and its details. He should be able to switch the current location or project or options.

3. Non-functional Requirements

This section describes the already known non-functional requirements for the product.

3.1 /NF10/Usability

The software should support a graphical user interface. It should be designed clearly and be easy to use, so that users can use the web application without difficulty even the first time they use it.

3.1.1 <UC.001>: Create PLC program

The user wants to create a graphical PLC program. He can choose between the programming languages FBD and SFC.

3.1.2 <UC.002>: Import file

The user wants to import an OpenPLC or AutomationML file to load the PLC-program.

3.1.3 <UC.003>: Export as PLCopen XML or AML

The user want to export the PLC program as PLCopen or AutomationML to exchange their program between development environments.

3.2 /NF20/License

The software shall be published under the MIT license. [1]

3.3 /NF30/System Environment

The editor shall be a browser application and shall be programmed with the help of the framework ANGULAR. It should work for the browsers (versions) Chrome 86.0.4240.183, Firefox 82.0.2 and Edge 85.0.564.63

4. Product Data

In the following, the product data is described.

4.1 /LD10/PLC

The editor shall support at least the PLC programming languages FBD and SFC.

4.2 /LD20/FB

The editor shall provide all current logic blocks as graphical library elements. These can be dragged into the editor window and connected to each other at the inputs and outputs.
Standard function blocks:

SR Bistable (SR) [4]
RS Bistable (RS) [4]
Semaphore (SEMA) [4]
Rising Trigger (R_TRIG) [4]
Falling Trigger (F_TRIG)[4]
Counter Up (CTU) [4]
Counter Down (CTD) [4]
Counter Up-Down (CTUD) [4]
Timer Pulse (TP) [4]
Timer On (TON) [4]
Timer Off (TOF) [4]

4.3 /LD30/File types

The system shall work with the file types, PLCopen XML and AML. It must transfer these files into a graphical PLC program or be able to export a PLC program in one of these file types. It's important, that the source document information exists and is filled in correctly. Otherwise, the transfer would not work.

4.3.1 PLCopen XML

The PLCopen XML (Extensible Markup Language) format contains the logical description and information about the different PLC programming languages and the program organization units (POU) created by the user within graphical information (like position and connection), functions, function blocks etc. [2]

4.3.2 AML

Automation Markup Language (AML) enables the XML-based representation of automation systems and objects through a transfer of technical data. [3]

5. References

[1] *https://opensource.org/licenses/MIT*
[2] *https://plcopen.org/*
[3] *https://www.automationml.org/*
[4] *https://www.openplcproject.com/plcopen-editor/*