

Customer Requirements Specification

(Lastenheft)

(TINF19C, SWE I Praxisprojekt 2020/2021)

Project: **PLCopen-Editor**

Customer: **Rentschler & Holder**
Rotebühlplatz 41
70178 Stuttgart

Supplier: Team 1 (Elian Yildirim, Franziska Kopp, Leonie de Santis, Mouaz Tabboush)
Rotebühlplatz 41
70178 Stuttgart

Version	Date	Author	Comment
0.1	16.10.2020	Leonie de Santis	Created
0.2	18.10.2020	Leonie de Santis	Added the goal and product environment
0.3	21.10.2020	Leonie de Santis	Use Cases created and completed with diagrams
0.4	24.10.2020	Leonie de Santis	Added Features and Business processes
0.5	28.10.2020	Leonie de Santis	Added Product Data and Other Product Characteristics

CONTENTS

1. Goal	3
2. Product Environment	4
3. Product Usage.....	5
3.1. Business Processes.....	5
3.1.1. <BP.001>: PLC programming	5
3.2. Use Cases	5
3.2.1. <UC.001> Create PLC program.....	5
3.2.2. <UC.002> Import File	6
3.2.3. <UC.003> Export File.....	7
3.3. Features.....	8
3.3.1. /LF10/ GUI Components	8
3.3.2. /LF20/ Editor	8
3.3.3. /LF30/ File validation	8
3.3.4. /LF40/ Define input/output Variables.....	8
3.3.5. /LF50/ Connections.....	8
3.3.6. /LF60/ Plausibility check	8
3.3.7. /LF70/ Exporter.....	8
3.3.8. /LF80/ Error handling.....	8
4. Product Data	9
4.1. /LD10/ PLC.....	9
4.2. /LD20/ FB	9
4.3. /LD30/ PLCopen XML	9
4.4. /LD40/ AML	9
5. Other Product Characteristics	10
5.1. /NF10/ Usability	10
5.2. /NF20/ License	10
5.3. System Environment	10
6. References	11

1. Goal

The goal of this project is to develop a web application which supports the commonly used programming languages of Programmable Logic Controllers (PLC) according to IEC 61131-3 [1]:

- Ladder Logic (LD)
- Function Block Diagram (FBD)
- Instruction List (IL)
- Structured Text (ST)
- Sequential Function Chart (SFC)

The editor of the open source project OpenPLC [2] can be used as a reference. The GUI shall be implemented as a browser based application based on ANGULAR. The editor shall enable the graphical creation of PLC programs and support at least FBD and SFC. All common logic blocks should be provided as graphical library elements which can be dragged into the editing window and linked with each other via their inputs/outputs. The programs should be able to be saved and loaded in the XML-format PLCopen[3] and AutomationML[4].

Optionally, a functional verification of a sample program on a runtime system and an interactive SCADA HMI on a suitable system without hardware I/O, can be performed.

2. Product Environment

The software is mainly used in automation technology for control and regulation tasks. The programmable logic controller (PLC) represents an open and universal automation solution for almost all industries and applications. The programming languages LD, FBD, IL, ST and SFC are to be supported. Initially, priority is given to the programming languages FBD and SFC.

SFC is a graphical representation for the description of sequential processes. It is often used in a superordinate way to break down a process into manageable units and to describe the control flow between these parts. The steps contained therein can be formulated independently of each other in another language defined in IEC 61131-3. FBD is another graphically oriented programming language for PLC. It works with a list of networks. Each network contains a structure that can contain logical and arithmetic expressions, function block calls, a jump or a return statement.

To exchange programs between development environments, the PLCopen XML and AutomationML formats are used. AutomationML (AML, Automation Markup Language) developed a neutral, XML-based and free standard for representing automation systems as objects. It enables a transfer of technical data. For this purpose already existing formats were adapted and merged in a suitable way. The representation of plant-specific data is possible in general and in particular the plant structure, geometry, kinematics and logical description. PLCopen XML is used for the logical description. PLCopen XML contains all important information about the different PLC programming languages, as well as the program organization units (POU) created by the user, such as functions and function blocks. It also includes graphical information such as the position and the connections.

3. Product Usage

The following business processes, use cases and features shall be supported by the system.

3.1 Business Processes

3.1.1 <BP.001>: PLC programming

Triggering Event:	The user opens the GUI and wants to create or upload a PLC program
Result:	The user creates a graphical PLC program. This can be created as FBD or SFC.
Involved Roles:	User and Editor GUI

3.2 Use cases

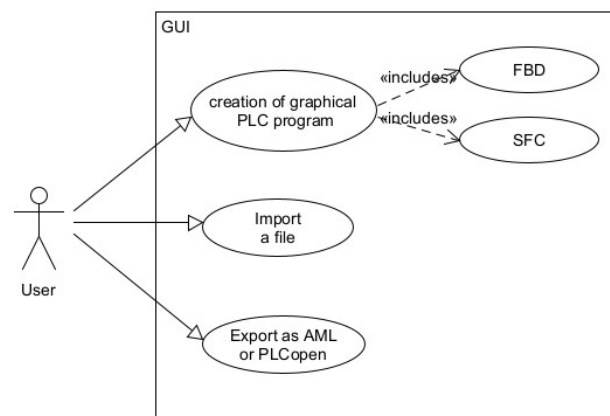


Figure 1: Use Case Overview Diagram

3.2.1 <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

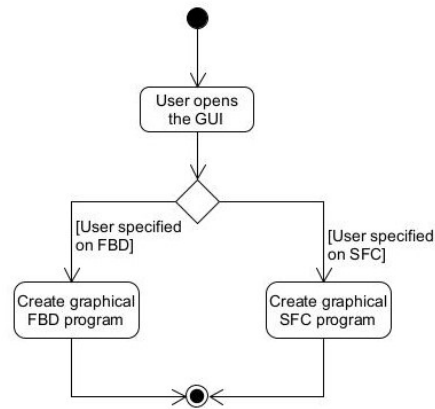


Figure 2: <UC.001> Create PLC program

3.2.2 <UC.002> Import file

Related Business Process:	<BP.001>: PLC programming
Use Cases Objective:	User wants to import an OpenPLC 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 GUI and want to import a file

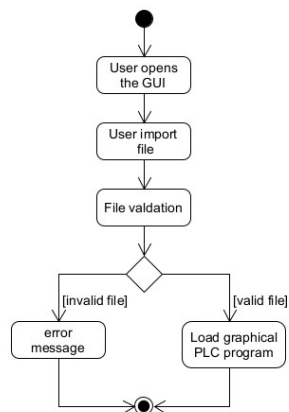


Figure 3: <UC.002> Import file

3.2.3 <UC.003> Export as PLCopen or AML

Related Business Process:	<BP.001>: PLC programming
Use Cases Objective:	User want to export the PLC program as PLCopen 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 finished the graphical PLC program and decide to save in PLCopen or AutomationML

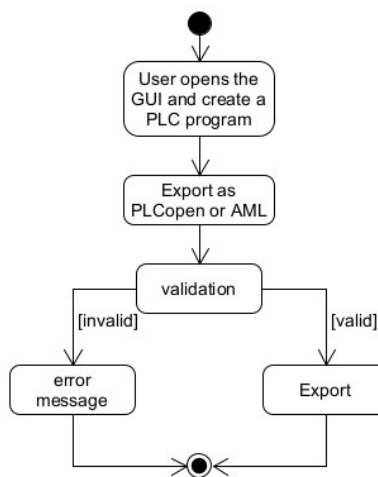


Figure 4: <UC.003> Export as PLCopen or AML

3.3 Features

The following functionalities shall be supported by the system.

3.3.1 /LF10/ GUI Components

The GUI has a homepage where a new project can be created or loaded. There should also be a project overview where new sections can be added to the project. These sections can then be edited in the editor and filled with blocks or corresponding modules.

3.3.2 /LF20/ Editor

The editor shall create a graphical PLC program from the imported file. If an error occurs, a corresponding error message should be displayed. If the file is imported completely, the user can edit the PLC program.

3.3.3 /LF30/ File validation

The File validation shall check if the imported file is convertible and corresponds to the PLCopen XML or AML schema. 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.

3.3.4 /LF40/ Define input/output Variable

The input and output variables of function blocks shall be defined.

3.3.5 /LF50/ Connections

The user interface offers through the intention of drag and drop functions a simple and clear editing and creation of the programs with blocks/respective modules. These blocks can be connected to each other at the inputs and outputs.

3.3.6 /LF60/ Plausibility check

The plausibility check is intended to check if the PLC program has a complete and correct structure. In case of violation, the user should be informed of his fault. Otherwise, the user can continue or export the program.

3.3.7 /LF70/ Exporter

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.

3.3.8 /LF80/ Error Display

The Software shall check if an error has occurred. If an error occurs, the user should receive an appropriate and understandable error message so that the error can be easily corrected.

4. Product Data

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.

4.3 /LD30/ PLCopen XML

The system shall work with PLCopen XML file. It must transfer these file into a graphical PLC program or be able to export a PLC program in one of these file types.

4.4 /LD40/ AML

The system shall work with an AML file. It must transfer these file into a graphical PLC program or be able to export a PLC program in one of these file types

5. Other Product Characteristics

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

5.1. /NF10/ Usability

The software should support a graphical user interface. It should be clearly laid out and easy to use so that the user can achieve the objectives effectively, efficiently and satisfactorily.

5.2. /NF20/ License

The software shall be published under the MIT license [5].

5.3. System Environment

The PLCopen-Editor is a browser-based application and therefore only requires an Internet connection and an installed web browser on your computer to work.

6. References

- [1] <http://tiegelkamp.eu/Buch/IEC61131-3JohnTiegelkampDeutschV1.2.pdf>
- [2] <https://www.openplcproject.com/plcopen-editor/>
- [3] <https://plcopen.org/>
- [4] <https://www.automationml.org/>
- [5] <https://opensource.org/licenses/MIT>