Customer Requirements Specification

(Lastenheft)

(TINF18C, SWE I Praxisprojekt 2019/2020)

Project:	*DD2AML Converter*
Customer:	Rentschler & Ewertz Rotebühlplatz 41 70178 Stuttgart
Supplier:	by Bastiane Storz - Team 3 (Nora Baitinger, Antonia Wermerskirch, Carl Beese, Lara Mack, Bastiane Storz) *Rotebühlplatz 41* *70178 Stuttgart*

Version	Date	Author	Comment
0.1	07.09.2019		created
0.2	04.10.2019	Bastiane Storz	Copy the goal from the task
0.3	21.10.2019	Lara Mack	Added BC/UC and Features, also added other product characteristics
0.4	22.10.2019	Bastiane Storz	BP translated into English and completed with diagrams, UC created and completed with diagrams
0.5	23.10.2019	Carl Beese	Goal and Product Environment translated; Product Data added
0.6	26.10.2019	Bastiane Storz	Fully translated, Product Environment dia-gram, generally verified
0.7	30.10.2019	Bastiane Storz	Improved on feedback
0.8	01.11.2019	Lara Mack	Improved on feedback, added two points on other product characteristics

Offene Punkte

CONTENTS

1. Goal

For the conversion of device description files for Profinet devices (called PN-GSD) to AML/CAEX V3, a tool called GSD2AML exist. GSDML is short for "General Station Description Markup Language" and AML is short for "Automation Markup Language". Both formats are based on XML.

The goal of this project is to develop a software which can convert other field-bus-specific formats to AML. Those formats will be IODD, ESI and CSP+ and the software should be able to recognize the given format on its own.

The generated AML-DD ("AML-Device-Description") must contain an AML root file with all the necessary information. The AML-DD with it's companion files shall be contained in an AMLX package which can be loaded into the AML editor.

The software shall be implemented as a library and supplied with a command line interface-based reference implementation and should also be usable trough a graphical user interface.

The documentation of the software as well as the documentation of the development and project is also part of the goal.

2. Product Environment

AutomationML (AML) is short for Automation MarkUp Language and is used to describe parts of automation plants as objects. These objects can consist of multiple other objects and be part of a larger assembly of objects. That way AML can be used to describe a single screw or an entire robot with the necessary level of detail.
AML makes use of various standards to describe plant components.

1. CAEX (Computer Aided Engineering Exchange) to describe attributes of objects and their relations in a hierarchical structure. This is called a system topology. In this respect, CAEX forms the overarching integration framework of AutomationML. [1]
2. COLLADA to describe the geometry and 3D models of a objects
3. COLLADA also integrates motion planning. It describes the connections and relations of moveable objects, which is called Kinematics
4. PLCopen XML describes the logic. Internal behavior and states if objects, action-sequences and I/O connections are implemented via this format.

An IODD (IO Device Description) file describes the sensors and actuator of a plant or component. It also contains information on Identity, parameters, process data, communication and more. It is written in XML-format, same as AML, which ensures a conversion. [2]

The ESI file format is used in networking hardware to specify information which is necessary for the use of the EtherCAT protocol.

CSP+ is used to describe components of the CC-Link Family or those which are compatible with them.

The converter will be implemented as a library. It will give the option of a standalone tool, a command line interface and a graphical user interface. The library will work with other software and thus supporting the export of an AML as a string.

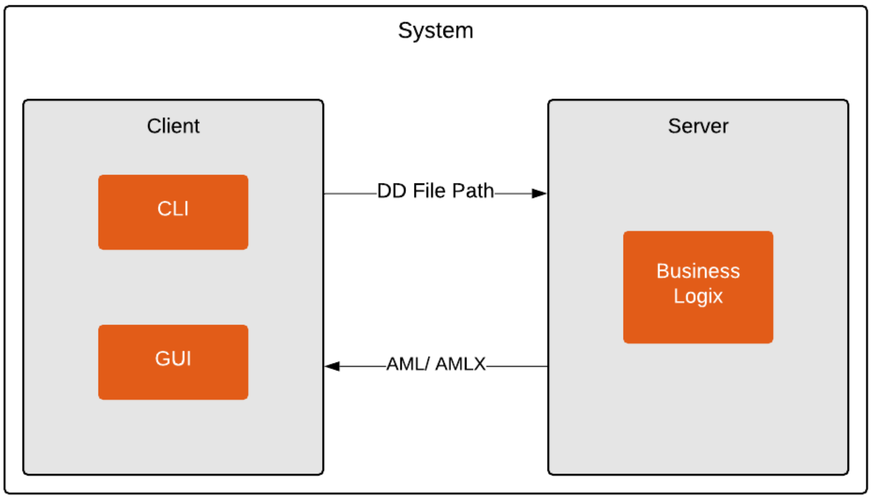

Figure 1 Product Environment

3. Product Usage

The main purpose of the software will be to convert the format into an AMLX file. Thus, from the user's point of view, the main case of application is the conversion into an AMLX file. This can be done using the options mentioned in the business process. To make things easier, the converted file should be stored in the same folder.

The following features must essentially be executable by the software:

1. The DD2AML converter automatically recognizes the input format.
2. Design and implementation of the DD2AML converter as command line tool and library. The tool should be implemented as a portable application without registry entries.
3. Transformation rules for IODD (IO-Link), ESI (EtherCAT) and CSP + (CC-Link) shall be developed analogous to the previous transformation rules for PN-GSDs (in gsd2aml.xml).

3.1. Business Processes

This section will make a difference between the conversion of the file and the implementation of the library in a different project.

3.1.1. <BP.001>: File conversion

Triggering Event:	User has a DD file and wants to convert it to an AMLX.
Result:	The system converts the DD file and stores the output AMLX file in the same directory.
Involved Roles:	User and DD2AML converter

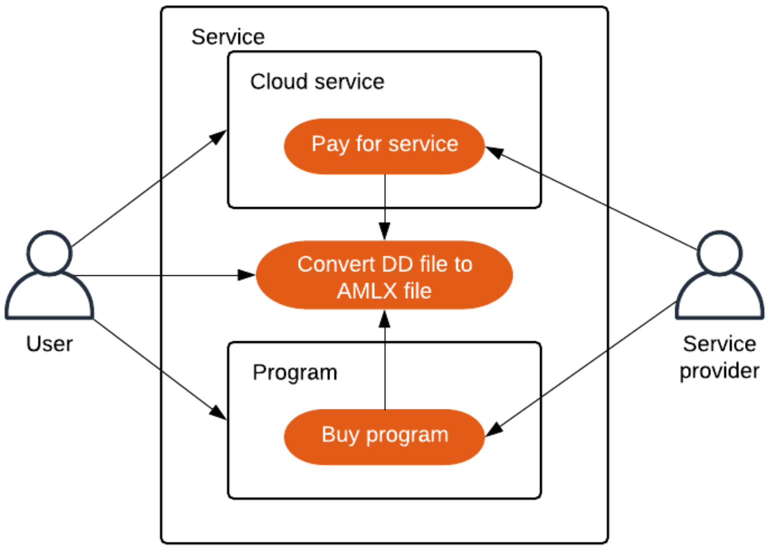

Figure 2 <BP.001> File conversion

3.1.2. <BP.002>: Library

Triggering Event:	Developer wants to implement the functionality of the DD2AML converter to his project.
Result:	The developer implements the library to his project.
Involved Roles:	Developer and DD2AML converter

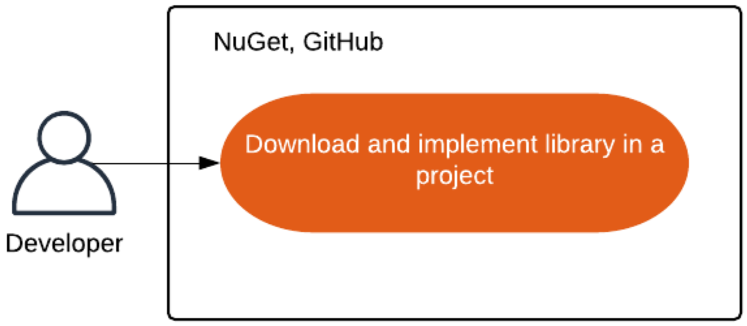

Figure 3 <BP.002> Library usage

3.2. Use Cases

This project will implement the library with a command line interface and optionally with a graphical user interface. Those will be two different, independent programs.
Secondly other developers might want to implement this library to their software. Since a developer might not want to save the file to the file system immediately, he can choose to get a string back from the library which contains the AML.

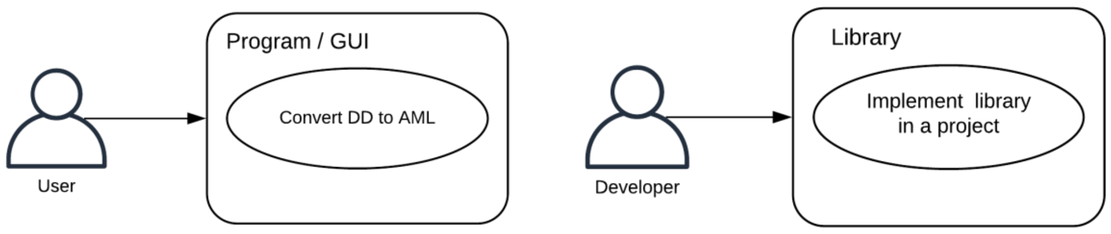

Figure 4 Use Case Overview Diagram

3.2.1. <UC.001> File conversion command line

Related Business Process:	<BP.001>: File conversion
Use Cases Objective:	User wants to convert his DD file to an AMLX file using the command line.
System Boundary:	Program itself is the system boundary.
Precondition:	- The program must be installed on the device. - "dd2aml" needs to be registered as a environment variable - The DD file must be without errors.
Postcondition on success:	The command line should not be closed before the conversion is finished.
Involved roles:	User and Converter
Triggering Event:	Typing dd2aml file with a valid FILEPATH in the command line

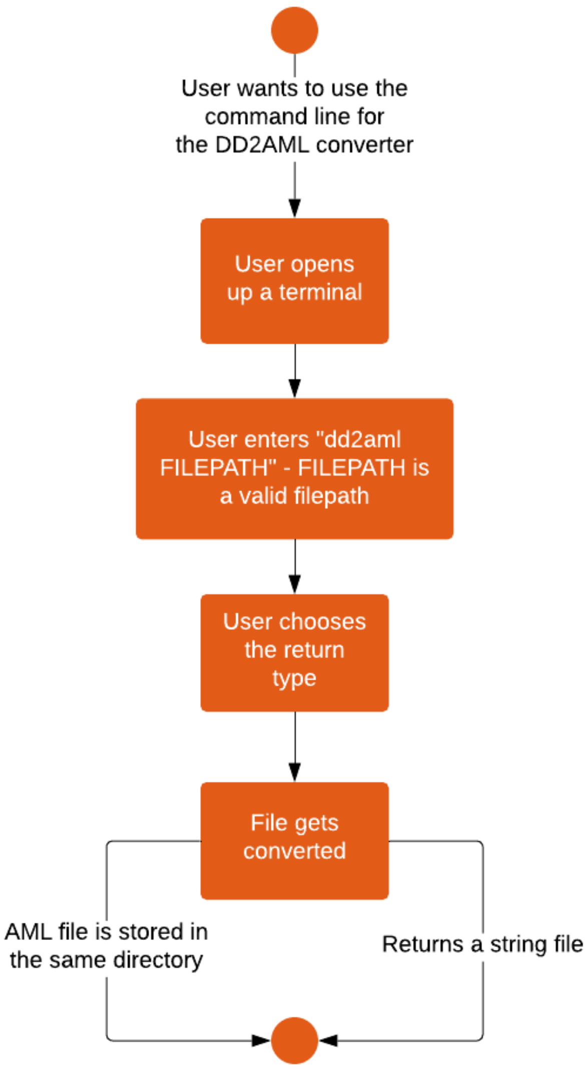
User wants to use the command line for the DD2AML converter
User opens up a terminal
User enters "dd2aml FILEPATH" - FILEPATH is a valid filepath
User chooses the return type
File gets converted
AML file is stored in the same directory
Returns a string file

Figure 5 <UC.001> File conversion command line

3.2.2. <UC.002> File conversion GUI

Related Business Process:	<BP.001>: File conversion
Use Cases Objective:	User wants to convert his DD file to an AMLX file using the GUI.
System Boundary:	Program itself is the system boundary.
Precondition:	- The program must be installed on the device. - The DD file must be without errors.
Postcondition on success:	The program should not be closed before the conversion is finished.
Involved roles:	User and Converter
Triggering Event:	Program is started.

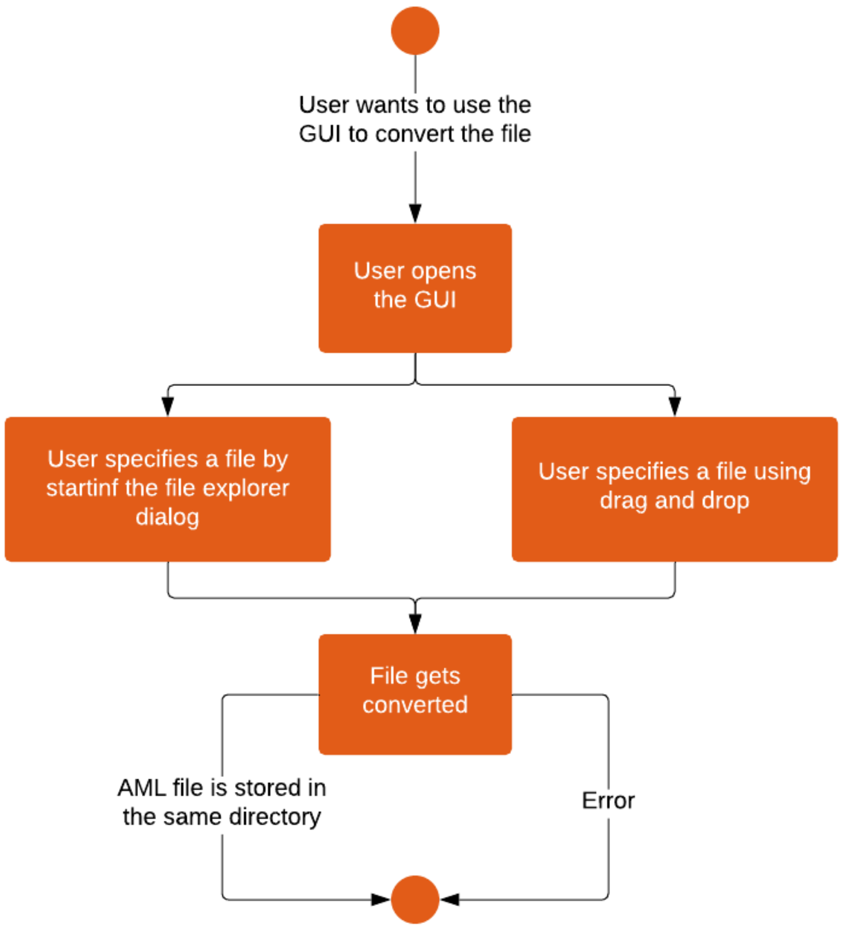
User wants to use the GUI to convert the file
User opens the GUI
User specifies a file by startinf the file explorer dialog
User specifies a file using drag and drop
File gets converted
AML file is stored in the same directory
Error

Figure 6 <UC.002> File conversion GUI

3.2.3. <UC.003> Library usage

Related Business Process:	<BP.002>: Library implementation
Use Cases Objective:	Possibility for developer to implement this project in their own software.
System Boundary:	Program itself is the system boundary.
Precondition:	The library needs to have a clearly documented interface and must be downloadable from GitHub or via NuGet. It needs to be published under the AGPL-license.
Postcondition on success:	The DD file was successfully converted into an AMLX package.
Involved roles:	Developer and library
Triggering Event:	Developer wants to implement a conversion function to his project.

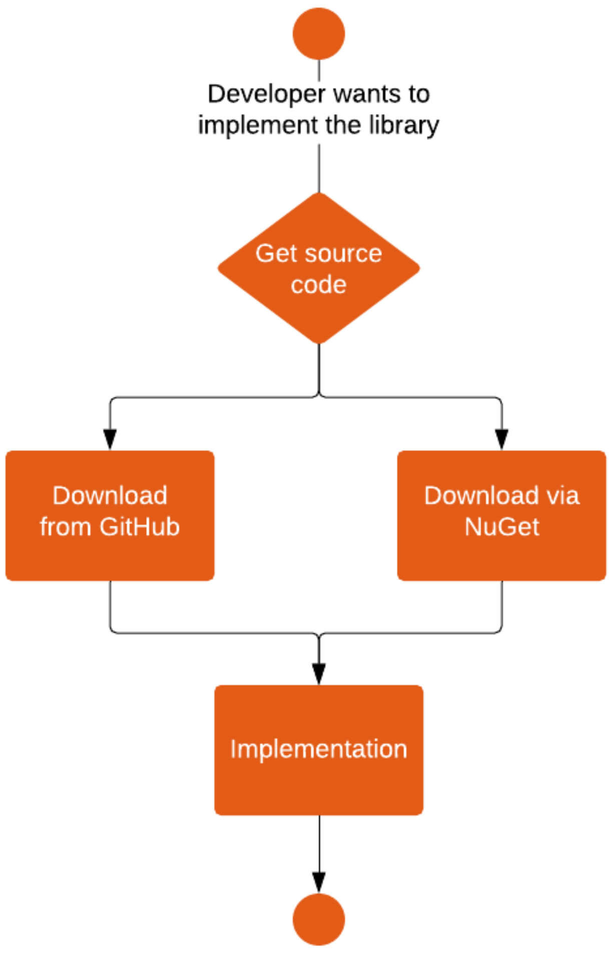

Figure 7 <UC.003> Library usage

3.3. Features

3.3.1. /F01/Import

The application should be able to import a file by the absolute path to the file.

3.3.2. /F02/Conversion

The system should be able to convert a DD file into an AML file.

3.3.3. /F03/Compressor

The system shall collect all resources and create an AMLX package from the AML file and the resource files.

3.3.4. /F04/Input validation

The system shall be able to detect wrongly formatted GSD files and throw an error to the user.

3.3.5. /F06/Error handling

The system shall be able to handle errors (unexpected shut down, wrongly formatted files, …) and throw an error to the user.

3.3.6. /F08/Output selection

The system shall provide the user with the ability to choose between storing the result as a file or returning the AML as a string.

4. Product Data

4.1. /LD10/ AML-DD

The system shall create a valid AML-DD with all the necessary information the original file contained. Including an AML root file complete with versioning header, a “SystemUnitClass” with logical description, identification and configuration parameters as well as a reference to the original file and pictures if there are any.

4.2. /LD20/ Original File

The system shall be able to convert a syntactically correct file of the formats IODD, ESI or CSP+.

4.3. /LD30/ AMLX package

The system shall create an AMLX file containing the AML-DD the original file and the device picture(s), which can be loaded into the AML editor.

4.4. /LD40/Command line interface

The software shall support a command line interface.

4.5. /LD50/Library

The software shall be implemented as a library so that other developers can use it in their projects.

5. Other Product Characteristics

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

5.1. /NF10/Graphical User Interface

The software should support a graphical user interface.

5.2. /NF20/System Environment

The software shall run under Windows 10 or lower version, because the only limitation is the .NET Framework.

5.3. /NF30/Installation wizard

The software shall be installable via an installer.
The installer shall allow to update to a newer version of the program.

5.4. /NF40/Portable Program

Also, the software shall exist as a portable program. This means that no installation is required.

6. Figures

7. References

[1] https://www.elektroniknet.de/automation/automationml-die-grundarchitektur-829-Seite-5.html
[2] https://www.kunbus.de/io-device-description.html