# System Architecture Specification

## *(Architekturspezifikation)*

(TINF18C, SWE I Praxisprojekt 2019/2020)

*Project:* *DD2AML Converter*

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

*Supplier:* by Nora Baitinger - Team 3
(Nora Baitinger, Antonia Wermerskirch, ~~Carl Beese~~, Lara Mack, Bastiane Storz)
Rotebühlplatz 41
70178 Stuttgart

| Version | Date | Author | Comment |
|---|---|---|---|
| 0.1 | 07.09.2018 | | created |
| 0.2 | 26.10.2019 | Bastiane Storz | Headings and table of contents revised and added technical concepts |
| 0.3 | 04.11.2019 | Carl Beese | Added System Overview, System Architecture, System Design |
| 0.4 | 05.11.2019 | Bastiane Storz | Design revised |
| 0.5 | 06.03.2020 | Bastiane Storz | Adjust to new requirements |
| 1.0 | 11.04.2020 | Nora Baitinger | Higher level of detail |

# Contents

# 1. Introduction

The goal of this project is to develop a software that supports the conversion from an IODD or CSP+ file to an AML file or respectively an AMLX package. The main part of this software should be a library that can perform such a conversion.
There also should be a command line tool and a tool with a graphical user interface, which use this library to convert DD files.

## 1.1. Glossar

| | |
|---|---|
| **.NET** | The .NET Framework is a software development and runtime environment developed by Microsoft for Microsoft Windows. |
| **AML** | Automation Markup Language is an open standard data format for storing and exchanging plant planning data. |
| **AML DD** | AML Device Description |
| **AMLX** | AML Package |
| **CLI** | The Command Line Interface from Microsoft Windows |
| **GSD** | General-Station-Description |
| **GUI** | Graphical User Interface |
| **CSP+** | The control and communication profile CSP+ is a specification and description file that contains and provides the necessary data for CC-Link family compatible devices for commissioning, operation and maintenance. |
| **IODD** | IO Device Description describes the sensors and other participants in an IO-Link network. |

# 2. System Overview

The system will work as follows: The user specifies a DD-file, the system checks the format of that file and validates the syntax. If the syntax is valid the system performs the conversion to the AML format, the result can either be saved as new AMLX package in the same directory or be returned as a string containing the AML file.

## 2.1. System Environment

There will be two ways to use the library which contains the transformation rules: As a stand-alone library or as a CLI or GUI Tool. Both require that the supplied DD-file is valid.
The first is the library which can be implemented into other projects. So other developers can use the conversion rules and use them for their own projects and define how the result should be used.
The second is the CLI or the GUI. They will give the user access to the library and the option to either create an AMLX package or just an AML file. Both can be loaded in the AutomationML editor.

## 2.2. Software Environment

The system requires the .NET framework version 4.7 and up in order to run. That framework only works on Windows 7 or later. The library can be implemented into any kind of software that utilities the .NET framework 4.7 or later.

## 2.3. Quality Goals

The following quality goals listed below should be achieved by the following architecture.

### 2.3.1. Usability

By offering different tools for different operating scenarios, a high degree of user-friendliness is achieved for every type of user. This means, developers can work with the conversion rules contained in the library and users can either work with the command line interface or when they prefer it, with the graphical user interface.

### 2.3.2. Maintainability

Dividing the software into smaller modules should help to make the software easier to analyse, maintain and modify. The result of dividing the software in smaller modules is shown in chapter 2.3 Quality Goals.

### 2.3.3. Portability

The software and the library should be portable. This means that the functionality to convert a DD file to AML/AMLX should be easily integrable by other software products. This is achieved by the three different use cases of the “DD2AML-Converter”.

# 3. Architectural Concept

The system will be based on previous efforts by a team of students who created something very similar, namely a converter for GSDML files.

## 3.1. Architectural Model

The system can be divided into two main parts. The execution layer, where the actual conversion takes place and the presentation layer, which makes the execution layer accessible for the user.

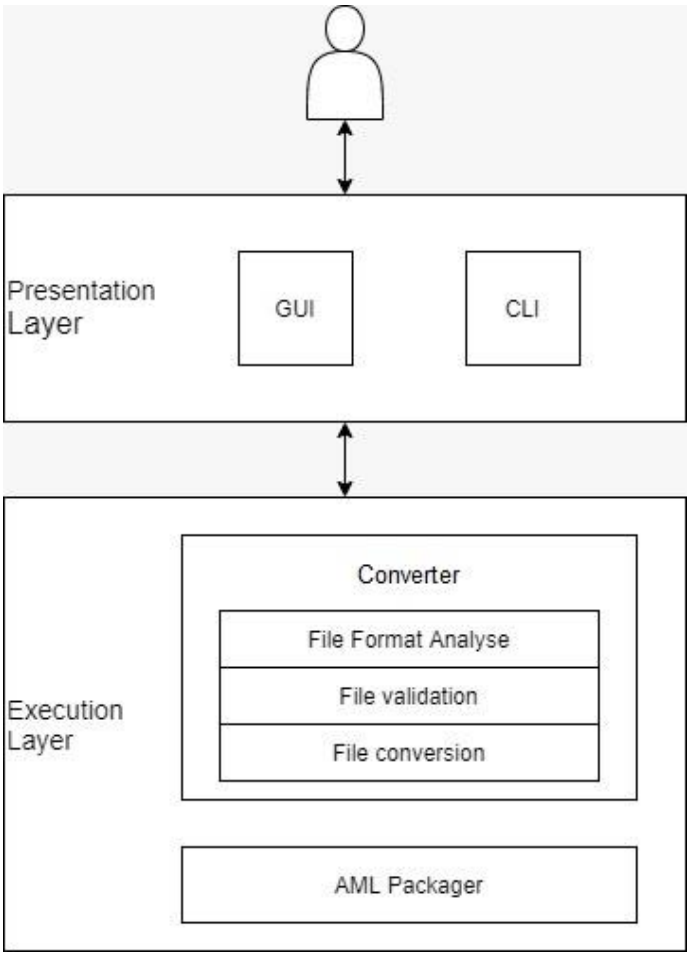


*Figure 1 - Architecture Model*

The presentation layer is divided into the two different ways to access the library. The library contains the conversion rules to generate the requested AML file.
The first option is the command line interface and the second the graphical user interface. The GUI will ensure the access to the library to users who are not familiar or comfortable with a CLI.
The second main section is the execution layer that can be divided into the two parts Converter and AML Packager. These two parts combined form the Library that can work as its own and could be used as a stand-alone library by developers.

## 3.2. Component Diagram

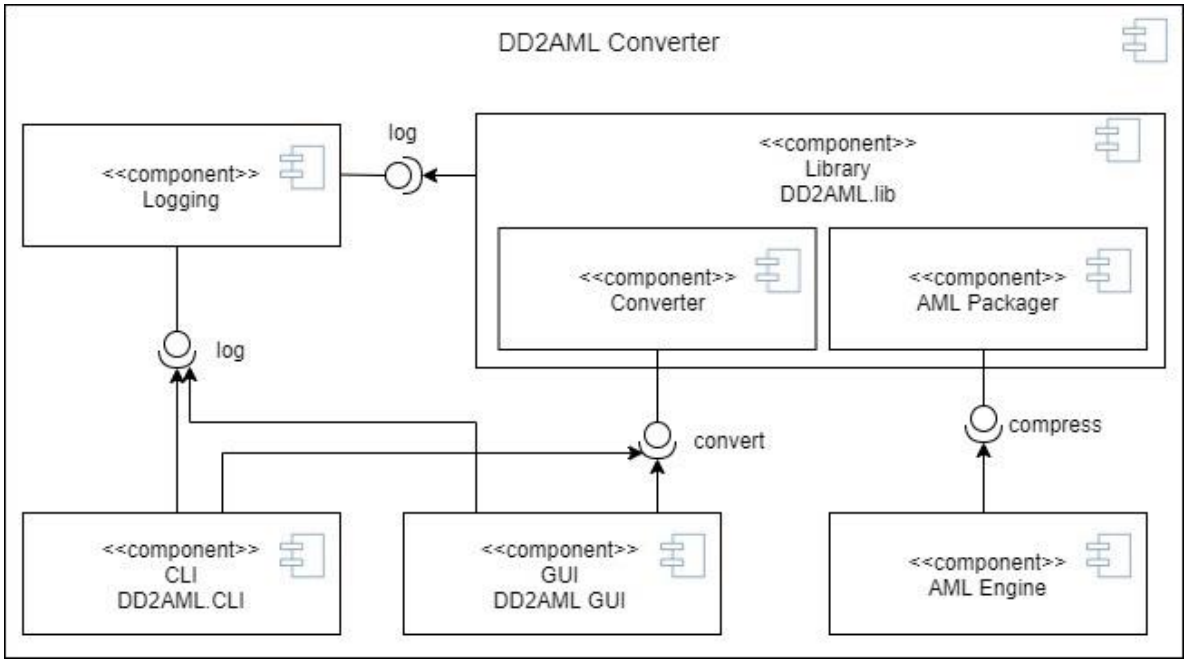


*Figure 2 - Component Diagram*

The main module of the whole DD2AML Converter is the Library. It is divided into two components. Together they can work as a stand-alone library to fulfil the task of converting a DD-file into the corresponding AML file and create the AMLX package.

The Converter module has some tasks to fulfil: First of all it must analyse the format of the given file and check if it is one of the listed ones (IODD, CSP+ and GSD).
It also has to validate the input file with a parser. In case the file is corrupted or contains invalid syntax, this will throw an exception and terminate the process.
The next task is the actual conversion. There will be specific conversion rules for each of the three initial formats to generate the correspondence to the newly generated AML file.

In the second part of the Library, the AML packer will create an AMLX package. It contains the logic to collect all the dependencies of the AML file and save them all to a new directory. It uses the AML Engine to get the task done.

The components CLI and GUI allow the user to access the library and to interact with the other components.

Additional, there will be a Logger module that provides a logging interface. It will be implemented in the other modules to log the errors and warnings that may occur.

# 4. Systemdesign

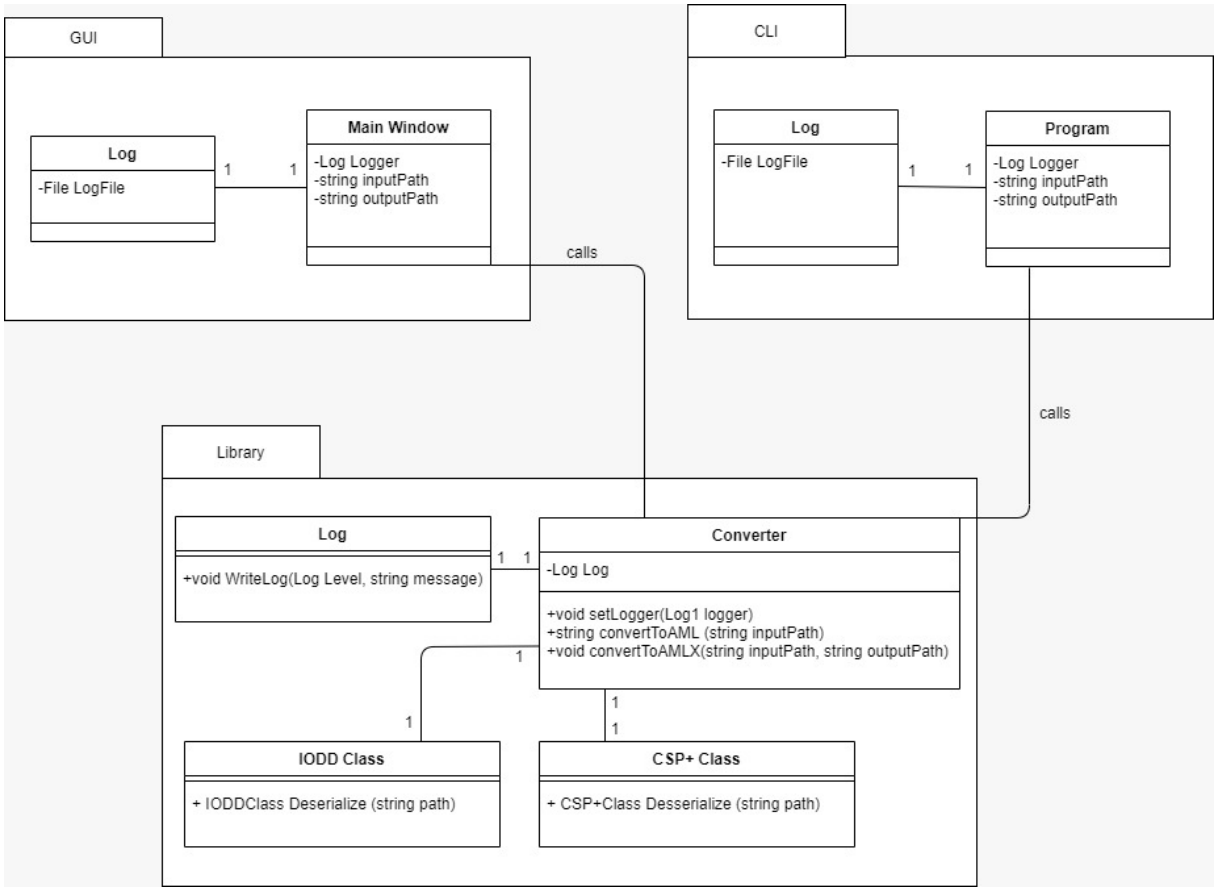


*Figure 2: System Design*

# 5. Subsystemspecification

## 5.1. <MOD.001>: Library

This Library is the most important module because it contains the logic for converting the input file to an AML file. It also includes the AML Packer that generates an AMLX with the AML file and all its dependencies.

| | |
|---|---|
| ***<MOD.001>*** | *DD2AML.lib* |
| ***System requirements covered:*** | */LF50/, /LF80/* |
| ***Service:*** | • *Providing a stand-alone library* |
| ***Interfaces:*** | • *Function that has the input format, the path to the input file and the path for the output AMLX package*<br>• *Function that has the input format, the path to the input file and returns an AML file as a string* |
| ***External Data:*** | • *Input file*<br>• *Output file / package* |
| ***Storage location:*** | *https://github.com/WAntonia/TINF18C_Team_3_DD2AML-Converter/tree/master/SOURCE/src/Dd2Aml.Lib* |

## 5.2. <SUBMOD.001.001>: Converter

This module performs the conversion. It takes the information from one of the format-classes and builds the AML accordingly.

| | |
|---|---|
| ***<SUBMOD.001.002>*** | *Converter logic* |
| ***System requirements covered:*** | */LF10/, /LF20/, /LF30/, /LF100/* |
| ***Service:*** | • *Handle incoming conversion requests*<br>• *Analyse the format type of the file*<br>• *Read and parse input file (file validation)*<br>• *Convert to AML file*<br>• *Return AML file* |
| ***Interfaces:*** | • *IODD class*<br>• *CSP+ class*<br>• *Output AML file structure* |
| ***External Data:*** | • *Input data from format-class*<br>• *Output AML file structure* |
| ***Storage location:*** | *https://github.com/WAntonia/TINF18C_Team_3_DD2AML-Converter/blob/master/SOURCE/src/Dd2Aml.Lib/Converter.cs* |

## 5.3. <SUBMOD.001.002>: AML Packager

The AML Packager collects all the dependencies and builds an AMLX package from the AML file and all additional referenced resources.

| ***<SUBMOD.001.003>:*** | *AML Packager* |
|---|---|
| ***System requirements covered:*** | */LF40/* |
| ***Service:*** | • *Collect all referenced resources*<br>• *Compress the collected data into AMLX package* |
| ***Interfaces:*** | • *File references in AML of input file* |
| ***External Data:*** | • *AML file*<br>• *Input file*<br>• *AMLX package* |
| ***Storage location:*** | *https://github.com/WAntonia/TINF18C_Team_3_DD2AML-Converter/blob/master/SOURCE/src/Dd2Aml.Lib/AMLPackager.cs* |

## 5.4. <MOD.002>: Command Line Tool

This module is about the command line tool. It specifies and implements the exact input and output for the command line interface.

| ***<MOD.003>*** | *Command Line Tool* |
|---|---|
| ***System requirements covered:*** | */LF60/, /LF100/* |
| ***Service:*** | • *Handle user input*<br>• *Display information to the user*<br>• *Handle all kinds of upcoming exceptions (Corrupted DD file, failed conversion, failed compression, …)* |
| ***Interfaces:*** | • *User input (Parameters, DD file path, AMLX output path…)*<br>• *Command line interface (Tool, provided by Windows)* |
| ***External Data:*** | • *Library*<br>• *Logging* |
| ***Storage location:*** | *https://github.com/WAntonia/TINF18C_Team_3_DD2AML-Converter/tree/master/SOURCE/src/Dd2Aml.CLI* |

## 5.5. <MOD.003>: Graphical User Interface

This module specifies and implements the graphical user interface and manages all possible in- and outputs.

| ***<MOD.004>*** | *Graphical User Interface* |
|---|---|
| ***System requirements covered:*** | */LF70/, /LF100/* |
| ***Service:*** | • *Display a graphical user interface to the user*<br>• *Handle user input*<br>• *Handle all possible outputs, including any kind of occurring exceptions* |
| ***Interfaces:*** | • *User input (DD file path, AMLX output path, trigger to start conversion)*<br>• *Self-defined graphical interface* |
| ***External Data:*** | • *Library*<br>• *Logging* |
| ***Storage location:*** | *https://github.com/WAntonia/TINF18C_Team_3_DD2AML-Converter/tree/master/SOURCE/src/Dd2Aml.Gui* |

## 5.6. <MOD.004>: Logging

The Logging module provides the logic for logging the systems status and the current state of conversion. This information will help the users to understand the converting process and achieve error information if necessary.

| ***<MOD.005>*** | Logging |
|---|---|
| ***System requirements covered:*** | */LF80/* |
| ***Service:*** | • *Specifying an interface for a logger that can be used within the library*<br>• *Implementing a logger based on the interface for the command line tool*<br>• *Implementing a logger based on the interface for the GUI tool* |
| ***Interfaces:*** | • *self-defined interface for logger* |
| ***External Data:*** | • *Log file* |
| ***Storage location:*** | • *https://github.com/WAntonia/TINF18C_Team_3_DD2AML-Converter/blob/master/SOURCE/src/Dd2Aml.CLI/Logger.cs*<br>• *https://github.com/WAntonia/TINF18C_Team_3_DD2AML-Converter/blob/master/SOURCE/src/Dd2Aml.Gui/Logger.cs*<br>• *https://github.com/WAntonia/TINF18C_Team_3_DD2AML-Converter/tree/master/SOURCE/src/Dd2Aml.Lib/Logging* |

# 6. Technical Concepts

## 6.1. Persistence

The goal of this project is to convert a file (GSD, IODD and CSP+) into an AML file. No other data is needed and therefore data persistence is not relevant for this kind of project.

## 6.2. User Interface

Users can access the conversion library via CLI or GUI. These two options will be specified in the module specification MOD.002 for the Command Line Tool and MOD.003 for the Graphical User Interface.

## 6.3. Ergonomics

For the Command Line Tool the ergonomics is not applicable.
The graphical user interface will follow the standard ergonomic design patterns. For example the font size should be large enough so the user experience is satisfying.

## 6.4. Communication with other IT-Systems

A developer can implement the library in their own software to interact with other IT-Systems. In addition to that users can use the library to convert their files via GUI or CLI, so no other communication is needed.

## 6.5. Deployment

The CLI tool and GUI tool will be deployed with an installation package (*.msi).

## 6.6. Data Validation

Before a conversion can take place, the input file (GSM, IODD and CSP+) needs to be validated to ensure a conversion is possible.

## 6.7. Exception Handling

When the input file is validated no other user interaction is needed for the conversion. Therefore no other exception handling is needed.

## 6.8. Logging

The logging will be specified by the logging module MOD.004.

## 6.9. Internationalisation

The language of the CLI and GUI is English. So are any needed information like the user manual and the readme files. For this reason the software can be used internationally.

## 6.10. Testability

The software is composed of different modules. These modules are tested separately. To receive an overview about the system tests the system test plan provides more information and the system test report contains all the results.

## 6.11. Availability

The program can be downloaded on GitHub and is therefore for everyone available.

# 7. Figures