Moduldokumentation

(MOD)

(TINF18C, SWE I Praxisprojekt 2019/2020)

Modul

Graphical User Interface

Project: *DD2AML Converter*

Customer: *Rentschler & Ewertz*
Rotebühlplatz 41
70178 Stuttgart

Supplier: by Bastiane Storz
- Team 3 (Nora Baitinger, Antonia Wermerskirch, Lara Mack, Bastiane Storz)

Rotebühlplatz 41
70178 Stuttgart

Version	Date	Author	Comment
0.1	07.09.2018		created
0.1	27.04.2020	Bastiane Storz	Filled with information
0.1	28.04.2020	Nora Baitinger	Checked
0.1	29.04.2020	Antonia Wermerskirch	Checked
0.2	03.05.2020	Bastiane Storz	Improved information
1.0	09.05.2020	Bastiane Storz	Improved design

1. Content

2. Scope

The Module Documentation (MOD) describes the architecture, the interfaces and the main features of the module. It also describes the module/component test including the results. It can also serve as a programming or integration manual for the module. If there are some risks related to the module itself, they shall be noted and commented within this document.

3. Definitions

GUI	Graphical User Interface
AMLX	AML Package
GSD	General Station Description
CSP+	Control and Communication System Profile
IODD	Input / Output Device Description
CAEX	Computer Aided Engineering Exchange
CLI	Command Line Interface

4. Figures

5. Module Requirements

5.1. User View

From the user point of view the GUI should complete the following tasks:

1. Offer the possibility to enter a GSD, CSP + or IODD file which the user wants to convert.
2. Offer the possibility to enter a file path in which the final AMLX package should be stored.
3. Give the user the chance that he can choose between CAEX Version 2.15 or 3.0.
4. Give the user the ability to start the conversion.
5. Informing the user about errors or a successful conversion.

5.2. Requirements

The functional requirements mentioned in the SRS LF10, LF70, LF90, LF100 and part of LF80 are implemented by the GUI.

According to LF10 the given file should be checked if it's one of the allowed file types. The user can only choose .xml or .cspp out of his explorer, because of that the chance that he chooses a wrong file is minimized. If the user wants to drop a file in the GUI he can only drag and drop files of the allowed filetypes into the textbox.

LF70 is fulfilled because the user gets a GUI where he can start the conversation and select things like the output directory, because according to the SRS the GUI shall provide the user with the necessary requests to start the conversion and give error information when needed.

LF90 is also given because it is possible to install the GUI with a given installation wizard like it said in the SRS, the software shall be installed using an installation wizard.

LF80 is saying, that the software shall be able to provide the user with understandable error messages. The user or any other developer working on the project should receive meaningful exception and error messages that help to identify an eventually occurring problem.

The user shall decide which version of the AutomationML / CAEX he or she wants to have. So, the software needs to give the opportunity to choose between the versions 2.15 and 3.0. This requirement is described in LF100 and will be realized in the GUI, where the user can switch with the help of a radio button between the versions.

5.3. **Module Context**

The module offers an interface to use the DD2AML converter for users, who are not familiar using the CLI.
The GUI needs all parameters, that are required by the converter library to process the conversion from the user and validate it. If all inputs are correct, it passes all information to the converter and start the conversion process. In this process the GUI gives the information to the converter module and works with the logging module.

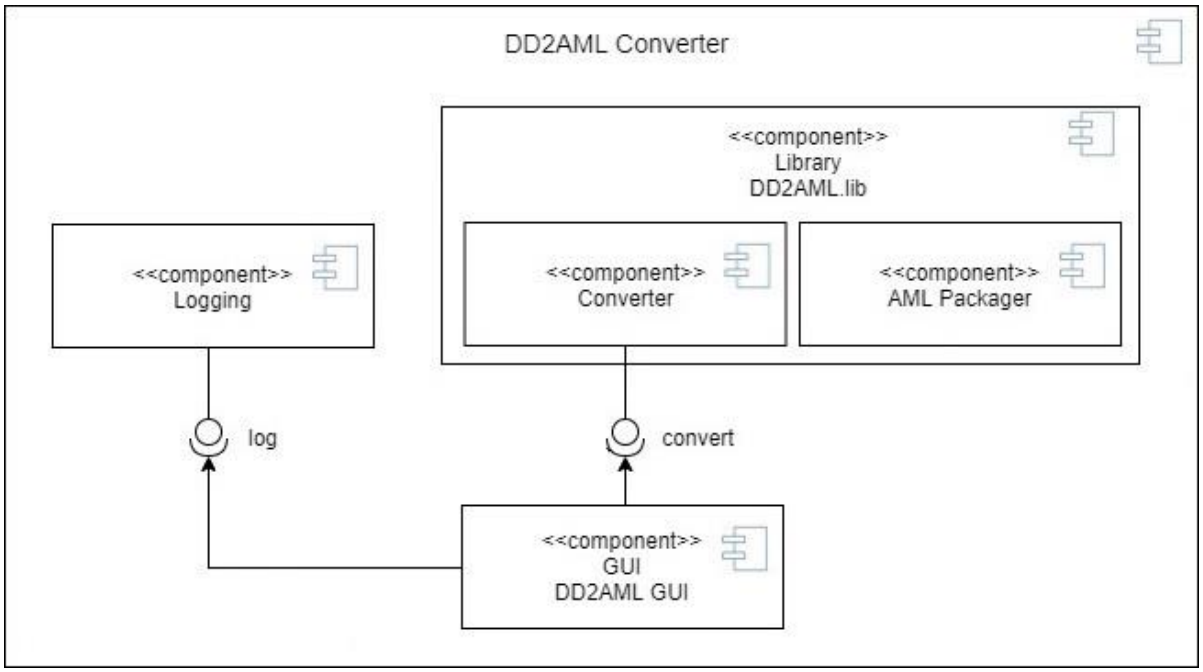

Figure 1 - Component diagram

6. Analysis

This module is all about providing a GUI to the user to perform a conversion form an GSD, CSP+ or IODD file to an ALMX package. This GUI must provide the user with the interaction capabilities described in chapter 4.

For the user to be able to give the required input, corresponding input options must be available. The required input results from the parameters required by the converter library to convert a GSD, CSP+ or IODD file. These parameters are the path to an existing GSD, CSP+ or IODD file, the path where to save the AMLX package, the information whether an existing file should be overwritten and the information whether the converter should check the input file validity.

In order to ensure that the converter receives correct input, this module must check if all user inputs are valid.

Furthermore, the user needs the ability to start the conversion and he must be informed of any problems that occur or of the completion of the conversion.

7. Design

All required functionalities of this module can be provided in a single dialog window, since only five direct inputs are required from the user. The just named inputs are the input field, output field, conversion button, strict mode and the possibility to choose different CAEX versions. This dialog window is implemented in the class called “MainWindow”.

This dialog box contains an input field that can be used to specify the input file to be converted. To the right of this input field there is a button which can be used to open an OpenFile Dialog. Optionally, the input file can also be specified via this dialog.
In order to make the GUI as user friendly as possible, it is also possible to just drag and drop a valid file into the window. Additionally, the output file for the AMLX package path will be automatically filled with a matched path to the given input file.

In the above-mentioned output field, you can specify the path where the AMLX package should be stored. Furthermore, there is also a button on the right end of the field which can be used to open a dialog for selecting a path, but it is not possible to drag files in here.
In both fields, input and output, there is a label in front of it to specify which is for what use case.

To start the conversion there is a button below the input fields. This button is disabled if the entries in the input field are not valid. As soon as there are valid, the button is enabled. When the button is pressed, the conversion process is started.

Below the conversion button there is a checkbox, which can be used to specify whether the converter should validate the input file. If this box is unchecked, the converter will not validate the given file.

On the right side of the checkbox there is another checkbox where the user can check if he wants the AMLX package in CAEX version 3.0, because the standard conversion version is 2.15.

For even more user friendliness, after a successful conversion the user will be asked, if he wants to open the generated AMLX package in the AutomationML Editor, when its installed on the same system.

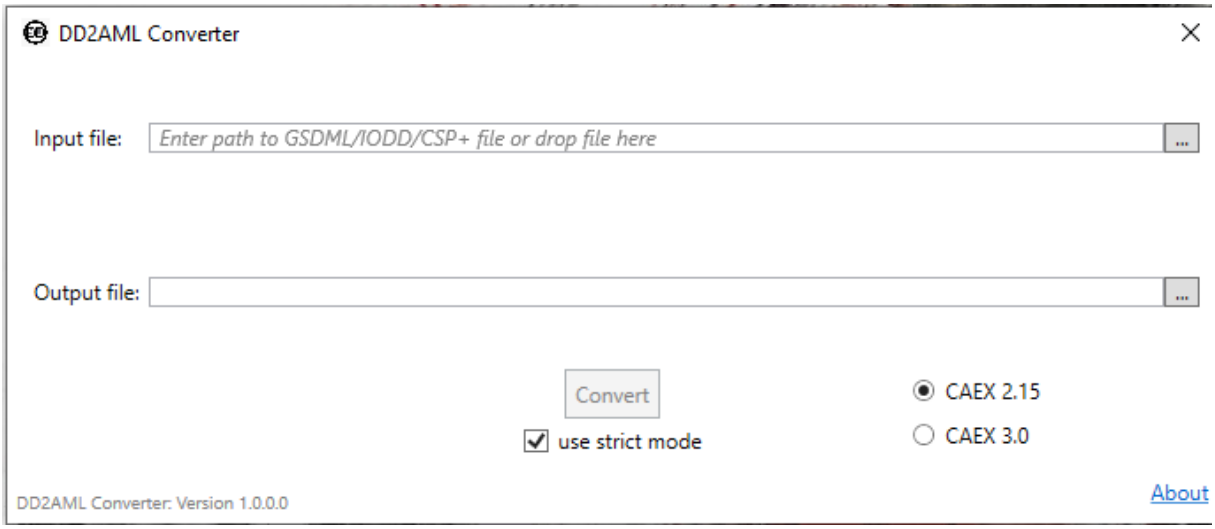

Figure 2 - DD2AML GUI

In order to guarantee these functionalities, the following functions and event handler must be implemented:

- "BrowseGsdFile_OnClick": An event handler that is triggered when the dialog button for selecting the GSD, CSP+ or IODD file is pressed. In this event handler the OpenFileDialog is opened to select the input file. If a valid selection is made, the selected path is written into the corresponding input field.

- "BrowseAmlFile_OnClick": An event handler that is triggered when the dialog button for selecting the path for the AMLX file is pressed. In this event handler the SaveFileDialog is opened to select the path. If a valid selection is made, the selected path is written into the corresponding input field.

- "TxtGsdFile_OnTextChanged": An event handler that is called when the text in the input field changes. Here the path to the AMLX package is set, if a valid input was made.

- "MainWindow_OnDragEnter": An event handler that is called when a file is dragged into the window. This is necessary to determine whether the file may be dropped.

- "MainWindow_OnPreviewDragOver": An event handler that is called when a file is moved across the window. This is necessary to determine whether the file may be dropped.

- "MainWindow_OnDrop": An event handler that is called when a file is dropped on the window. Here the path of the dropped file is read and written into the input field.

- "TxtGsdFile_OnDragEnter": An event handler that is called when a file is dragged into the input field. This is necessary to determine whether the file may be dropped.

- "TxtGsdFile_OnPreviewDragOver": An event handler that is called when a file is moved across the input field. This is necessary to determine whether the file may be dropped.

- “TxtGsdFile_OnDrop”: An event handler that is called when a file is dropped on the input field. Here the path of the dropped file is read and written into the input field.

- “Convert_OnClick”: An event handler that is triggered when the button to start the conversion is pressed. In this event handler the conversion library is called, the parameters are passed, and the conversion is started. The errors that can occur during the conversion are also caught here and displayed to the user via a dialog box.

- “GetAmlEditor”: A helper function that searches for an installed AutomationML editor and returns the path to it if it is found.

The functionality described above can be found in the library path of the project: “[source]/src/Dd2Aml.Gui/MainWindow.xaml.cs“.
All log files can be found in “C:/Users/[your user]/AppData/Local/DD2AML/Logs/[GUI | CLI]/”.

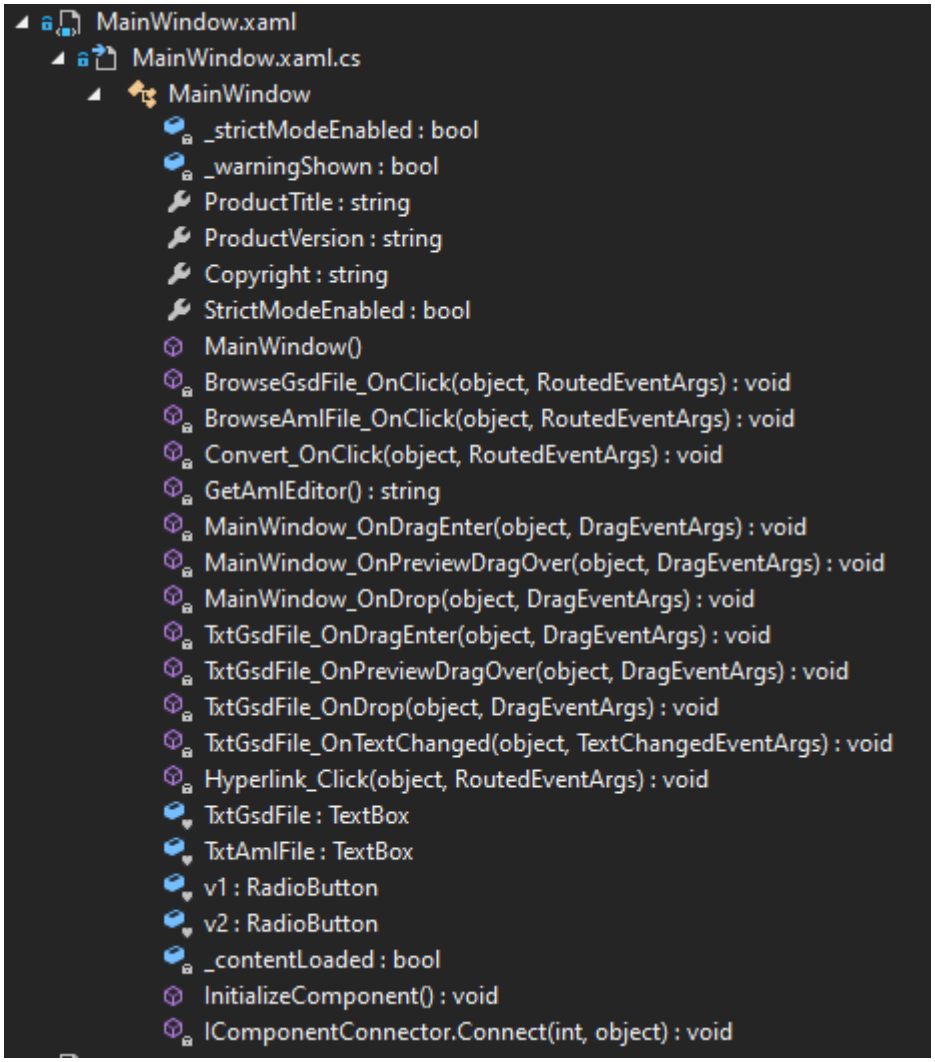

Figure 3 - MainWindow.xaml.cs - Function Overview

7.1. **Risks**

Even if the GUI display is based on the .NET framework, which is maintained and tested by Microsoft, it cannot be completely excluded that the application is displayed incorrectly. If the display is incorrect, the application may no longer be usable.

As a 3rd party the WIX Tool and the extension for Visual Studio 2019 is used. It provides us the possibility to make an installer for the GUI because visual studio doesn't support this function anymore. This is kind of a risk because when the installer should be changed the WIX Tool always need to be available even in the future.

8. Implementation

Since all the required functionality was known beforehand and is described in chapter 6 the module simply implements all the listed functions. For that, the .Net IO functionality was used for interacting with the file system and path handling.
All the functions were put on visibility private except for the “Compress” method which will be the entry point for other modules and programs using this library.
An end to end test was implemented to ensure that the process is working.
The code is documented using C#XML documentation comments. These are machine readable and can therefore be used by different tools to provide useful information to the user of the library.

9. Module Test

Automated Unit tests are not possible for the GUI, because there are interactions with the user needed.
Instead it can be looked to the Module Test Cases in chapter 10.4.

10. Summary

The module implements the requirements completely. It is sufficiently tested and deals with all errors by raising them to the caller or by using the logger module to inform the user.

11. Appendix

11.1. References

[1] SRS: *https://github.com/WAntonia/TINF18C_Team_3_DD2AML-Converter/wiki/System-Requirements-Specification*

[2] STP: *https://github.com/WAntonia/TINF18C_Team_3_DD2AML-Converter/wiki/Systemtestplan*

[3] STR: *https://github.com/WAntonia/TINF18C_Team_3_DD2AML-Converter/wiki/Systemtestreport*

11.2. Code

The source code for this module can be found at:

- https://github.com/WAntonia/TINF18C_Team_3_DD2AML-Converter/tree/master/SOURCE/src/Dd2Aml.Gui

11.3. Module Test Cases

Testcase ID:	TC-003-001	
Testcase Name:	GUI Input field verification	
Req.-ID:	LF70	
Description:	Run converter via graphical user interface with an empty Input field. The test ensures that a conversion is not possible without an input file.	
Test Steps		
Step	**Action**	**Expected result**
1	Install the DD2AML Software and open the GUI.	The software is installed and the GUI window opens.
2	Try to start the conversion by pressing the “Convert” button at the bottom centre.	Conversion not possible, because "Convert" button stays deactivated.
Testcase ID:	TC-003-002	
Testcase Name:	GUI Input file selection via file explorer	
Req.-ID:	LF70	
Description:	The test verifies that only the permitted file formats can be selected as input via file explorer. Permitted file formats: .xml and .cspp	
Test Steps		
Step	**Action**	**Expected result**
1	Install the DD2AML Software and open the GUI.	The software is installed and the GUI window opens.

2	Click on the "..." button at the end of the input text field.	The file explorer opens in a new window.
3	Click on "Files" in the lower right corner directly above the buttons for open and cancel	A drop-down menu opens showing that only file suffix with .xml or .cspp are allowed.

Testcase ID:	TC-003-003
Testcase Name:	GUI Input file selection via drag and drop
Req.-ID:	LF70
Description:	The test verifies that only the permitted file formats can be selected as input via drag and drop Permitted file formats: .xml and .cspp

Test Steps

Step	Action	Expected result
1	Install the DD2AML Software and open the GUI.	The software is installed and the GUI window opens.
2	Open the file explorer and select any file. Drag the selected file and drop it into the GUI input text field.	If the selected file has a valid file suffix, its absolute file path will appear in the input field. If it has an invalid suffix, it is not possible to drop the file into the input field.

Testcase ID:	TC-003-004
Testcase Name:	GUI Output file path generation
Req.-ID:	LF70
Description:	The test verifies whether an output file is automatically suggested for a given input. This output file should no longer have the file format of the input file in its name.

Test Steps

Step	Action	Expected result
1	Install the DD2AML Software and open the GUI.	The software is installed and the GUI window opens.
2	Select a valid file of IODD, CSP+ or GSD format in the Input text box.	As soon as the file including file path is in the input field, an output file is suggested for the same directory. The output file has the suffix. amlx and does not have the file format of the input file in its name.