Moduldocumentation

(MOD)

(TINF20C, SWE I Praxisprojekt 2021/2022)

Modul

Graphical User Interface

Project: *Modelling Wizard for Devices*

Customer: *Rentschler & Holder*
Rotebühlplatz 41
70178 Stuttgart

Supplier: by Lukas Ernst – Team 1
(Linus Eickhoff, Florian Kellermann, Lukas Ernst, Malte Horst, Florian Kaiser)
Rotebühlplatz 41
70178 Stuttgart

Version	Date	Author	Comment
V0.1	07.09.2021	Lukas Ernst	Created
V0.2	27.04.2022	Lukas Ernst	Filled with information
V0.3	28.04.2022	Florian Keller-mann	Improved information
V0.4	03.05.2022	Linus Eickhoff	Checked
V1.0	06.05.2022	Lukas Ernst	Improved design

Contents

1. Scope

This module documentation explains the GUI in more detail. It shows how the Standalone Application is now graphically structured and which features are implemented. The individual functions are tested in advance and their results are documented here. If there are existing problems, they are also listed here and possible solutions are explained in more detail.

It can also serve as a programming guide, if further features should be implemented.

2. Definitions

GUI - Graphical User Interface
SRS - System Requirement Specification
STP - System Test Plan
STR - System Test Report
AMLX - AML Package
CAEX - Computer Aided Engineering Exchange

3. Figures

4. Module Requirements

4.1. User View

This Module should provide the user the following features:

1. Open the plugin without editor, in a separate window
2. A better graphical experience to the previous project
    - Using the entire space of the window
    - Non-resizable rows (cause it is redundant)
    - In general the design of all elements e.g. colors, shape, size...
3. Link to the Manual

4.2. Requirements

The following requirements are implemented by this module: /LF10/, /LF30/, /LF40/, /LF50/, /LF60/, /LF70/, /LF80/, LD/20/.

/LF10/: The user now has the possibility to import files using an absolute path. In the beginning this did not work anymore.

/LF30/: The system now detects incorrectly formatted files and displays an error message.

/LF30/, /LF40/, /LF50/: These requirements are primarily just visual changes that should improve the usability of the GUI and give the user a better experience with the plugin. The main focus here is to make sure that all the space is used, the colors are rearranged and that the buttons can be used more intuitively. All the graphical changes have been implemented in the meantime. Other improvements (Enhancements) have also been incorporated into the program to further increase usability.

/LF60/: When the user is shown the attributes of a loaded device, he can edit any attribute he wants to change.

/LF70/: When starting the application, the user can create a new, empty device model.

/LF80/: When the user has edited a device, the device can save to a file.

4.3. Module Context

This module provides a graphical interface to the Standalone Application. By adding interfaces, filling and editing the attributes and attaching images and icons, the graphical interface should give the user a simplified way to create files. It is attempted to make the interface as practical as possible. To make this possible, for example, the standard class with the basic information is automatically loaded and the mandatory attributes are simple to find. If all information is correct, it will be passed to the controller, which will process the entered information.

5. Analysis

The graphical user interface must provide the user with a simple, intuitive way to create files. This requires the ability for the user to interactively enter information and be supported by the graphical user interface. This support could be, for example, the possibility to drag libraries into the desired field or also to specify which attributes must be filled in. It must be checked whether the user has entered all the required data. If this is not the case, an appropriate and specific error message must be returned to the user, so that he knows what he must do to improve this error. If other problems occur because the user does something unexpected or something that is requested by the program, but corrupts the export of the file, a specific error message must also be returned to the user.

6. Design

The graphical user interface from the previous project was divided into three tabs. (Figure 1, 2, 3)

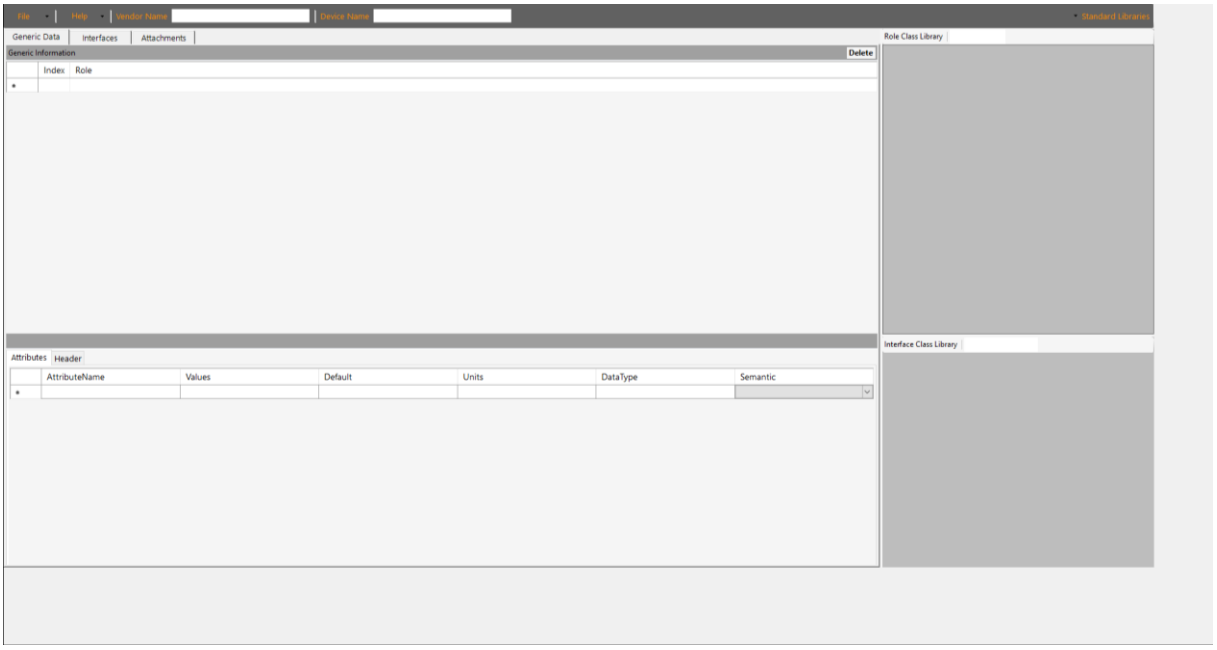

Figure 1 – Start Screen Generic Data

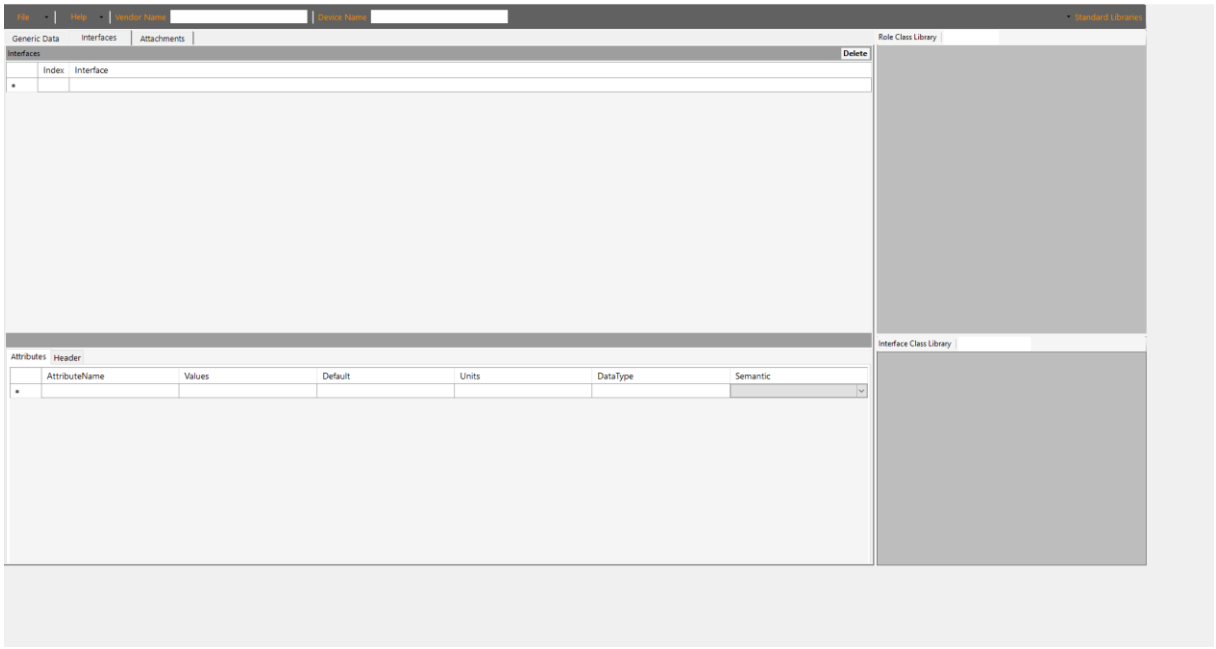

Figure 2 – Interface Tab

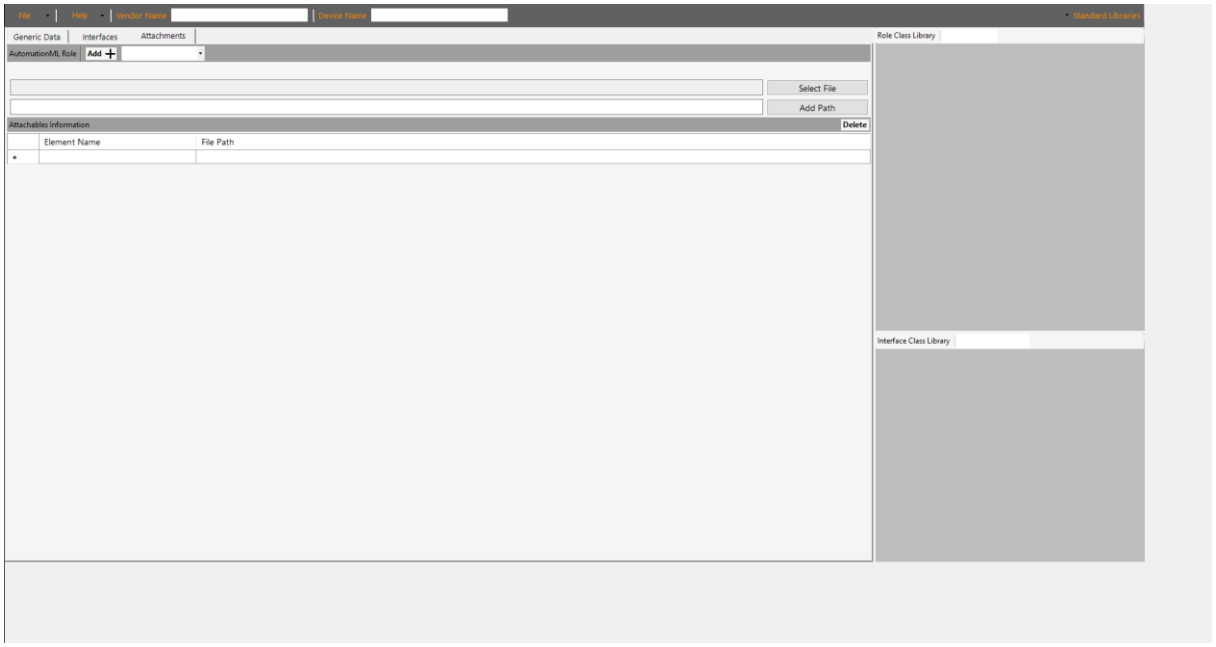

Figure 3 – Attachments Tab

The Generic Data Tab contains the Role Classes, which are important to store general information. The most important data is also stored there, such as the name of the device and the manufacturer name. In the Interfaces Tab interfaces are added, this could be for example an electrical interface that defines the pins of a USB socket in more detail. In the last tab it is possible to add further attachments to the file. This was the current state of the previous version. Our task is to completely renew the design, because the interface was only partially usable. There were a lot of errors when interacting with the objects, objects that covered other objects and the design didn't look so good. However, we made sure that the look and colors of each tab were consistent.

6.1. **Risks**

The graphical user interface was created using the .NET for Windows framework, more specifically the GUI toolkit called Windows Forms, which is tested and maintained by Microsoft. The risk is that the development depends on the utilizability of the toolkit. If Microsoft no longer supports the maintenance of Windows Forms there could be complications displaying the Plugin. As a result, it cannot be excluded that the display will work with 100% accuracy.

A small, but still noteworthy risk are shapes that are not available, which thereby cannot be implemented. For example, buttons are linked to certain design-patterns. So it is not possible to customize the button as the developer wants. However, this is only a small risk, since WindowsForms offers quite a lot of design freedom.

7. Implementation

The graphical user interface is implemented with the GUI toolkit WindowsForm as already mentioned in chapter 6.1. This toolkit allows changes to be made directly in a graphical interface in visual studio. You can simply change the properties of an object. It is also possible to bind functions to specific events for example, when the user clicks on a specific object. This way the functions are bound from the Controller-Module to the GUI. Creating objects and modifying their properties creates automatically generated code, which is stored in the DeviceDescription.Designer.cs file. This also makes it possible to modify the code the way the developer prefers it. This means that the developer is not bound to the graphical interface, but can also modify the code manually. All the requirements related to this module have been implemented with WindowsForms.

8. Module Test

In this section nearly all requirements will be tested separately on their functionality. Some requirements are not tested, because they have no function to test on.

8.1. Module Testplan

Req. - ID	Functionality
LF10: Import	Imports file by absolute path
LF30: Error handling	Application throws errors on expected shutdowns and wrong formatting
LF40: GUI	Draws GUI for user
LF60: Edit device	Every attribute of devices should be editable
LF70: Create device	Creates a new and empty device
LF80: Export device	Loaded device is saved as to file

8.2. Module Testreport

Req. - ID	Pass/ Fail	When failed: Observation	
LF10: Import	Pass		Linus Eickhoff
LF30: Error handling	Pass		Linus Eickhoff
LF40: GUI	Fail	File is not displayed as expected.	Florian Kaiser
LF60: Edit device	Pass		Malte Horst
LF70: Create device	Pass		Linus Eickhoff
LF80: Export device	Fail	File is saved and exported correctly without errors when creating a new de-vice. While editing an ex-isting device, exporting fails.	Florian Kaiser

9. Summary

This module was successfully implemented with all requirements. The GUI now has a new design that is more intuitive, user-friendly and cleaner. By testing the GUI, some bugs have appeared, which have been additionally fixed to make the user experience even better. Each error is handled by notifying the user with a specific message to clarify what problem has occured.

10. Appendix

10.1. References

[1] System Requirements Specification: https://github.com/H4CK3R-01/TINF20C_ModellingWizard_Devices/wiki/1.-Software-Requirements--Specification
[2] Previous Project: https://github.com/DekaAthlos/TINF19C-ModellingWizard
[3] System Test Plan: https://github.com/H4CK3R-01/TINF20C_ModellingWizard_Devices/blob/47d2ba67fc73ebc080f303f0e29ca2260d8c7d88/PROJECT/STP/TINF20C_STP_Team_1.pdf
[4] System Test Report: https://github.com/H4CK3R-01/TINF20C_ModellingWizard_Devices/blob/47d2ba67fc73ebc080f303f0e29ca2260d8c7d88/PROJECT/STR/TINF20C_STR_Team_1.pdf

10.2. Code

The source code for this module can be found at:

- https://github.com/H4CK3R-01/TINF20C_ModellingWizard_Devices/blob/app-source-code/SOURCE/Application/Properties/Resources.Designer.cs