Duale Hochschule Baden-Württemberg

TINF21C, SWE I Praxisprojekt 2022

Customer Requirements Specification (Lastenheft)

Project: **Modelling Wizard Improvments**

Customer: Rentschler & Holder
Rotebühlplatz 41
70178 Stuttgart

Supplier: Team 4 (Dana Frey, Michael Grote, Nils Hoffmann, Fabian Kreuzer, Maximilian Trumpp, Robin Ziegler)
Rotebühlplatz 41
70178 Stuttgart

Version 0.1

Document author	Maximilian Trumpp	Created on	15.10.2022
Filename	TINF21C_CRS_Team_4_0v1.doc		
Number of pages	X	Duale Hochschule Baden-Württemberg	

Version History

Version	Date	Author	Comment
0.1	15.10.2022	Maximilian Trumpp	Created first version
0.2	XX.10.2022	(Vorname Name)	
0.3	XX.10.2022	(Vorname Name)	

Table of contents

1 Goal

The Goal is to revise the hole application. The focus should be on the revision of the UI and on the improve of the usability of the app. To improve the user-friendly of the app, more parts of the easy mode should be moved to the advanced mode. Future more some functions which are available over two menu points, with different options, should be bundled into one menu point. The normal user just wants to create or import an existing device and change single values and not wants to create a complex device. For a better usability its necessary to refractor the hole code, with the refactoring of the code the existing bugs needs to be fixed. In the case of an error the customers wished to be better informed and if possible, to directly jump with the cursor into the field, where the error was triggered.

2 Product Environment

AutomationML (AML) is the short form of Automation MarkUp Language and is used to describe parts of automation plants as objects. These objects can consist of multiple other objects and can 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 the following 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.
2. COLLADA to describe the geometry and 3D Models of an object
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 sensor and actuator of a plant or component. It also contains information on identify, parameters, process data, communication and more. It is written in XML-format, same as AML, which ensures a conversion.

3 Product Usage

The following business processes, use cases and features shall be supported by the system. The processes are splatted into easy-mode process and advanced mode process.

3.1 Business Process

3.1.1 <BP.001> Create, Edit, Save and Export Devices

Triggered Event:	User wants to create a new device, edit an existing device and save or export a device.
Result:	The user can open, create, edit and save or export a device with the application.
Involved Roles:	User and Device Modelling Application

3.1.2 <BP.002> Create and Load Interfaces from Library

Triggered Event:	The user wants to create a new interface or load interfaces from a library
Result:	The user can add interfaces to device files and is able to import interfaces into the application.
Involved Roles:	User and Device Modelling Application

3.2 Use Cases

3.2.1 <UC.001> Create device in easy mode

Related Business Process:	<BP.001> User wants to create a new device file in easy mode.
Use Cases objective:	User in easy mode wants to create a device by inserting the data manually into the user interface of the application.
System Boundary:	The application itself.
Precondition:	The user needs to know the minimal required parameters for the device. The program needs to be opened on the user's system.
Postcondition on success:	The entered data is displaced completely and correctly. The user needs to easily find where to edit which parameter.
Contributed User:	Every end-user of the application in easy mode.
Triggering Event:	When the user opens the application and uses the 'new device' button to create a new device in easy mode.

3.2.2 <UC.002> Create device in advanced mode

Related Business Process:	<BP.001> User wants to create a new device file in advanced mode
Use Cases objective:	User in advanced mode wants to create a device by inserting the data manually into the user interface of the application and by manually adding device interfaces.
System Boundary:	The application itself.
Precondition:	The user needs to know the minimal required parameters for the device. If it's a complex device, the user needs to know which interfaces are included. The program needs to be opened on the user's system.
Postcondition on success:	The entered data is displaced completely and correctly. The user needs to easily find where to edit which parameter or where to add or remove interfaces.
Contributed User:	Every end-user of the application in advanced mode.
Triggering Event:	When the user opens the application and uses the 'new device' button to create a new device in advanced mode.

3.2.3 <UC.003> Edit device in easy mode

Related Business Process:	<BP.001> User wants to edit a device file in easy mode
Use Cases objective:	User in easy mode wants to edit a device by inserting the changed data manually into the user interface of the application.
System Boundary:	The application itself.
Precondition:	The program needs to be opened on the user's system and the file needs to be opened.
Postcondition on success:	The entered data is displayed completely and correctly. The user needs to easily find where to edit which parameter.
Contributed User:	Every end-user of the application in easy mode.
Triggering Event:	When the user opens the application and opens a file by using the 'open' button in easy mode.

3.2.4 <UC.004> Edit device in advanced mode

Related Business Process:	<BP.001> User wants to edit a device file in advanced mode
Use Cases objective:	User in advanced mode wants to edit a device by inserting the changed data manually into the user interface of the application and by manually adding or removing interfaces.
System Boundary:	The application itself.
Precondition:	The user needs to know the minimal required parameters for the device. The program needs to be opened on the user's system.
Postcondition on success:	The entered data is displaced completely and correctly. The user needs to easily find where to edit which parameter or where to add or remove interfaces.
Contributed User:	Every end-user of the application in advanced mode.
Triggering Event:	When the user opens the application and opens a file by using the 'open' button in advanced mode.

3.2.5 <UC.004> Export device

Related Business Process:	<BP.001> User wants to export a device file.
Use Cases objective:	The user wants to export a created or edited device.
System Boundary:	The application itself.
Precondition:	The user has created/added or loaded a device.
Postcondition on success:	The entered data is displayed completely and correctly. The user has one device opened.
Contributed User:	Every end-user of the application in advanced mode.
Triggering Event:	When the user uses the ‘export’ button to export a device.

3.2.6 <UC.004> Format output as CAEX version 2.15/3.0

Related Business Process:	<BP.001> User wants to create a new device file in advanced mode
Use Cases objective:	The user wants to save a created or edited device.
System Boundary:	The application itself.
Precondition:	The user has created/added or loaded a device.
Postcondition on success:	The entered data is displayed completely and correctly. The user has one device opened.
Contributed User:	Every end-user of the application in advanced mode.
Triggering Event:	When the user uses the ‛save button to save a device.

3.2.7 <UC.004> Load interface library

Related Business Process:	<BP.002> Load interfaces
Use Cases objective:	The user wants to import an existing library.
System Boundary:	The application itself.
Precondition:	The user needs a correct library file. The program needs to be opened on the user's system.
Postcondition on success:	The entered library is displayed completely and correctly. All imported interfaces should be displayed.
Contributed User:	Every end-user of the application in advanced mode.
Triggering Event:	When the user opens the application and uses the 'import interface' button to import a new library.

3.2.8 <UC.005> Add attachments for the device

Related Business Process:	<BP.001> User wants to create a new device file in advanced mode
Use Cases objective:	User wants to add an attachment to an opened device.
System Boundary:	The application itself.
Precondition:	The user needs an attachment. The program needs to be opened on the user's system.
Postcondition on success:	The attachment is attached to the device.
Contributed User:	Every end-user of the application.
Triggering Event:	When the user opens the application and uses the 'Add Attachment' button to add an attachment.

3.3 Features

3.3.1 /Feature/OtherFileName

The file name should be built from the product code instead of the device name.

3.3.2 /Feature/ReplaceDragAndDrop

Interface shouldn't be added by drag&drop, if one interface from the library is touched it should be added.

3.3.3 /Feature/ReplaceOptionsForOpen

The options for open a file should be replaced.

3.3.4 /Feature/ShowAllDataModels

The tree view with the interfaces should be expanded automatically.

3.3.5 /Bug/FileOpen

Fixing a bug where the datamodel disappears after opening a file.

3.3.6 /Feature/EaslierEasymode

With the user is in the easy mode there shouldn't be shown any interfaces on the right. If the user is in easy mode there should just be an option for import, edit, save and export devices. It should be just possible to edit already included interfaces. For editing interfaces, the user needs to switch into the advanced mode.

3.3.7 /Feature/ErrorHandeling

If there appears an error while saving the file, because of a missing attribute the curser of the user should automatically jump into the missing field.

3.3.8 /Feature/RemoveExtraInput

The existing fields in top right should be removed. The name and producer attributes should just be editable in the attribute table on the button.

3.3.9 /Feature/MoveSaveFunction

The two existing functions 'Save' and 'Save & Close' should be moved into one function which is named 'Save'.

3.3.10 /Bugfix/SaveFunction

After using the button 'Save & Close' the application can't be used anymore.

3.3.11 /Feature/Import

The two existing import functions should be moved into one single function, where the file type can be chosen from all available file types.

3.3.12 /Bugfix/OpenSavedFile

Fixing a bug where a saved file can't be opened anymore. Happened with '.amxl', also test others.

4 Production Data

4.1 /Feature/GUIImprovments

The existing GUI should be refactored to look more modern

4.2 /Feature/Usability

The hole application should be checked and refactored for a better usability.

5 Other Product Characteristics

6 Prototype of the UI

6.1 Prototype of the UI in easy mode

6.2 Prototype of the UI in advanced mode