DHBW
Duale Hochschule
Baden-Württemberg
Stuttgart

Customer Requirements Specification

(TINF21C, SWE)

Project: Modelling Wizard Improvements

Customer: Markus Rentschler

Christian Holder

Team:

Project Manager	– Robin Ziegler (inf21100@lehre.dhbw-stuttgart.de)
Developer	– Nils Hoffmann (inf21194@lehre.dhbw-stuttgart.de)
Test Manager	– Michael Grote (inf21111@lehre.dhbw-stuttgart.de)
System Architect	– Fabian Kreuzer (inf21106@lehre.dhbw-stuttgart.de)
Tech. Documentation	– Dana Frey (inf21099@lehre.dhbw-stuttgart.de)
Product Manager	– Maximilian Trumpp (inf21123@lehre.dhbw-stuttgart.de)
Sophie Kirschner	– Graphical Designer (inf21983@lehre.dhbw-stuttgart.de)

Change History

Version	Date	Author	Comment
0.1	20.09.2023	Maximilian Trumpp	Created first version
0.2	23.09.2022	Maximilian Trumpp	Edited Use Cases
0.3	30.09.2022	Maximilian Trumpp	Edited Features
0.4	06.10.2022	Maximilian Trumpp	Added Prototype
0.5	04.04.2022	Michael Grote	Edited missing numbers of Features
0.6	14.04.2023	Maximilian Trumpp	Revision
1.0	11.05.2023	Maximilian Trumpp	Final version

Main author and responsible for this document: Maximilian Trumpp

Table of Contents

1. Introduction

This document is the basis of the Modelling Wizard project of the TINF21C team. The document contains all requirements of the customer for this project. The other documents use this document as a basis.

2. The objective

The goal of this project is to revise the application. The focus should be on revising the user interface and improving the usability of the application. To improve the usability of the app, more parts of the simple mode should be moved to the advanced mode. The menu of the application should be revised. Menu items with similar functions are to be grouped together for a better overview. According to the requirements, the normal user would like to create or import only an existing device and change individual values and not create a complex device. For a better usability it is necessary to refactor the whole code, with the refactoring of the code the existing bugs have to be fixed. In case of an error, customers wished to be better informed and, if possible, to jump directly with the cursor to the field where the error was triggered.

3. 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.

4. Product Usage

4.1 Business Process

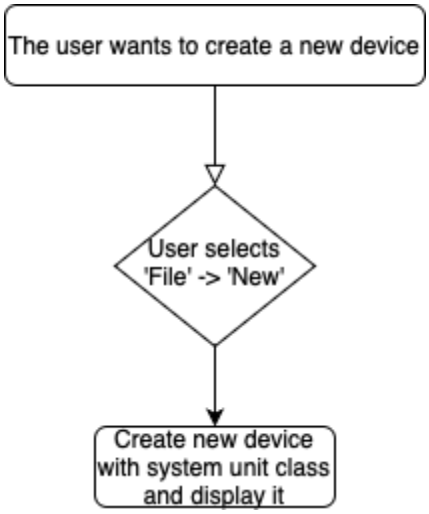
All Business Processes are included in the old wiki of the application. Due to the requirement to fix the bugs and improve usability, no new business processes have emerged.

4.2 Use Cases

The project should not receive any new use cases, as this project is only about usability improvements.

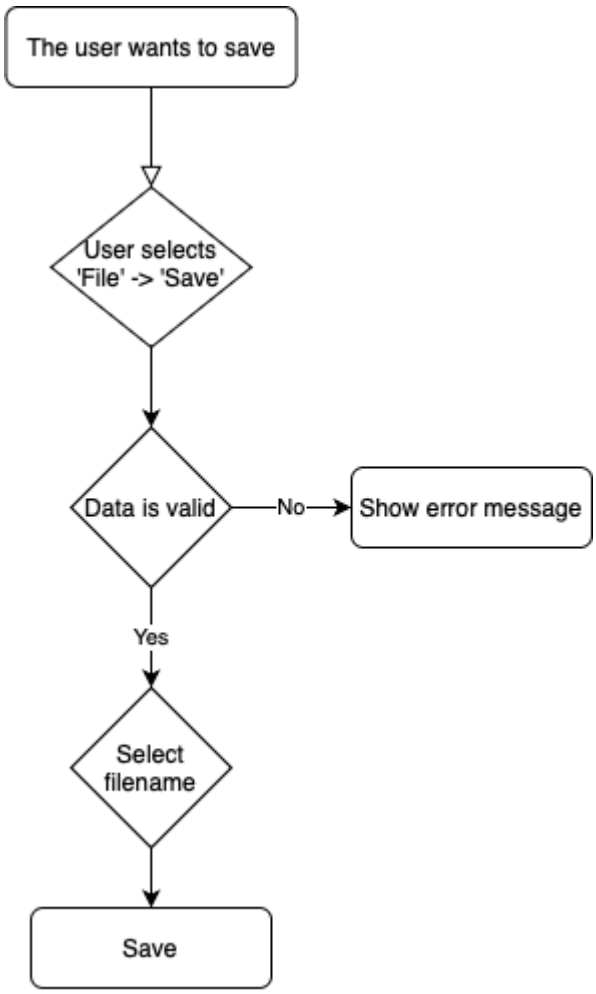
4.2.1 <UC.001> Create new device

Use Cases Objective	User 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 user needs to know the minimal required parameters for the device. The program needs to be opened on the user's system.
Contributed user	Every end-user of the application.
Triggering event	When the user opens the application and uses the 'new device' button.



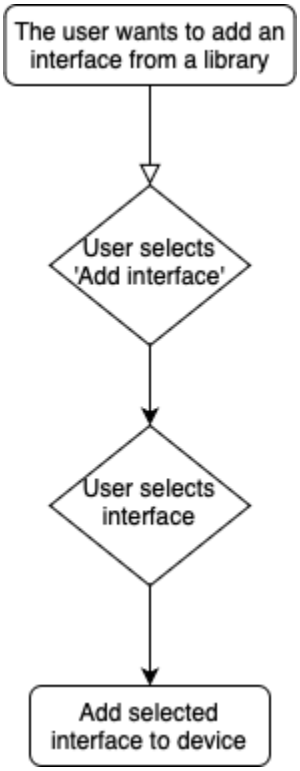
4.2.2 <UC.002> Save device

Use Cases Objective	The user wants to save a currently opened device, by using the "Save" button.
System Boundary	The application itself
Precondition	The user created a new device or has loaded an existing device from a file.
Postcondition on success	The data from the program is saved correctly.
Contributed user	Every end-user of the application.
Triggering event	When the user clicks the "Save" button.



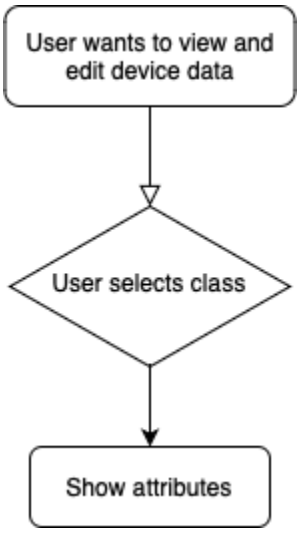
4.2.3 <UC.003> Add interface from existing library

Use Cases Objective	User wants to add an existing interface from the loaded library.
System Boundary	The application itself
Precondition	The user needs to know the minimal required data for the interface to be added.
Postcondition on success	The user has successfully added an interface from the loaded library to the loaded device.
Contributed user	Every end-user of the application.
Triggering event	When the user clicks on the "Add Interface" button.



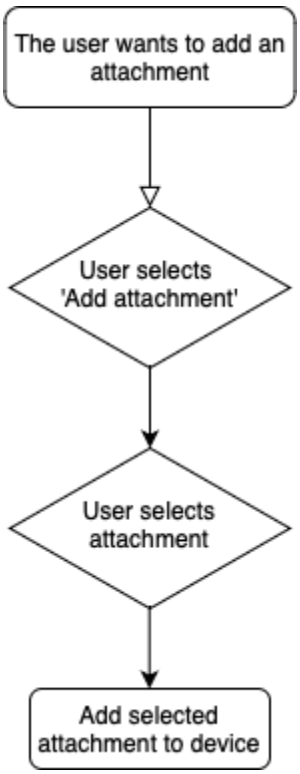
4.2.4 <UC.004> View and edit device data

Use Cases Objective	When a device is either opened or created, the user should see all device data. The user should also be able to edit them.
System Boundary	The application itself
Precondition	Valid device informations were loaded.
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.
Triggering event	A device was successfully opened or created from an existing file.



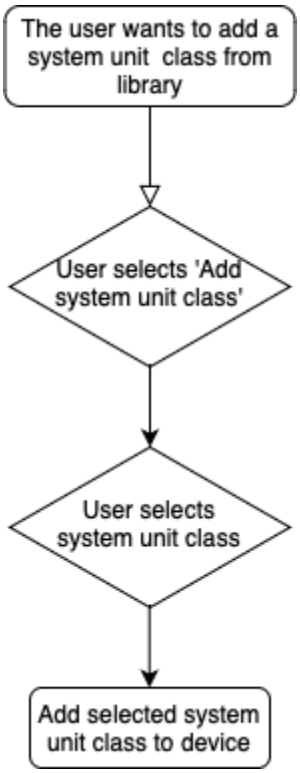
4.2.5 <UC.005> Add attachments to device

Use Cases Objective	The user wants to add an attachment to a device.
System Boundary	The application itself
Precondition	The user has either loaded an existing device or created a new one.
Postcondition on success	The user has successfully added a system unit class to the loaded device.
Contributed user	Every end-user of the application.
Triggering event	When the user uses the "Add Attachment" button.



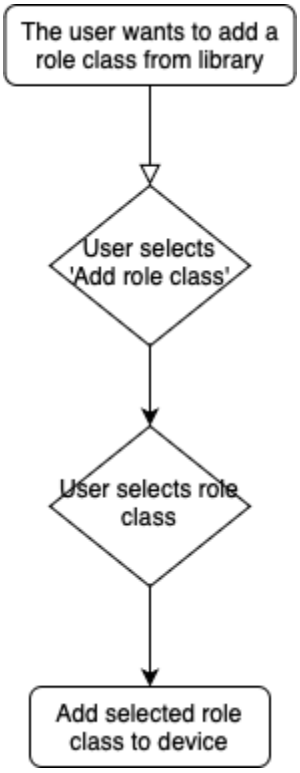
4.2.6 <UC.006> Add system unit classes to device

Use Cases Objective	User wants to add an existing system unit class from the loaded library.
System Boundary	The application itself
Precondition	The user needs to know the minimal required data for the system unit class to be added.
Postcondition on success	The user has successfully added a system unit class from the loaded library to the loaded device.
Contributed user	Every end-user of the application.
Triggering event	When the user clicks on the "Add System Unit Class" button.



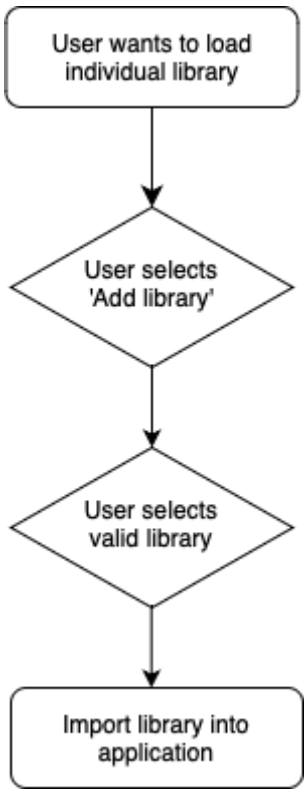
4.2.7 <UC.007> Add role class to device

Use Cases Objective	User wants to add an existing role class from the loaded library.
System Boundary	The application itself
Precondition	The user needs to know the minimal required data for the role class to be added.
Postcondition on success	The user has successfully added a role class from the loaded library to the loaded device.
Contributed user	Every end-user of the application.
Triggering event	When the user clicks on the "Add Role Class" button.



4.2.8 <UC.008> Load individual library

Use Cases Objective	User wants to use an individual library file.
System Boundary	The application itself.
Precondition	The user needs a valid library file.
Postcondition on success	The individual library can be used to add classes.
Contributed user	Every end-user of the application.
Triggering event	When the user clicks on the "Add library" button in the menu.



4.3 Functional Requirements

In this chapter, the definable features are described and illustrated with figures.

4.3.1 /LF11/Import

The application should be able to import a file by the absolute path to the file. Supported filetypes should be AMLX, AML, EDZ, IODD and GSD.

4.3.2 /LF12/File Validation

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

4.3.3 /LF13/Error Handling

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

4.3.4 /LF14/Edit device

The system should display of a device after either load or open a device. The user should be able to edit displayed informations.

4.3.5 /LF15/Create device

When the user creates a new device, it should not have any values except for the System Unit class.

4.3.6 /LF16/ Export device

After the end of the editor the user should have the possibility to save the device in a file.

4.3.7 /LF17/ Easy Mode

The user wants to be able to view and edit basic information of the attributes.

4.3.8 /LF18/ Expert Mode

The user wants to be able to view and edit advanced information of the attributes.

4.4 Non-functional Requirements

4.4.1 /NF11/GUI improvements

The application should present an appealing interface. The graphical user interface should have a very good usability for an easy operation.The user should be able to navigate easily and intuitively in the application.

4.4.2 /NF12/Display device in a readable way

The informations from the either opened or created device should be displayed in a readable way for the user. In expert mode the user can view more details of the device.

4.4.3 /NF13/Portable

If possible, the application should be able to run as a portable, without an installation.

4.4.4 /NF14/Performance

The application should respond instantly after a user's action. There should not be long loading times.

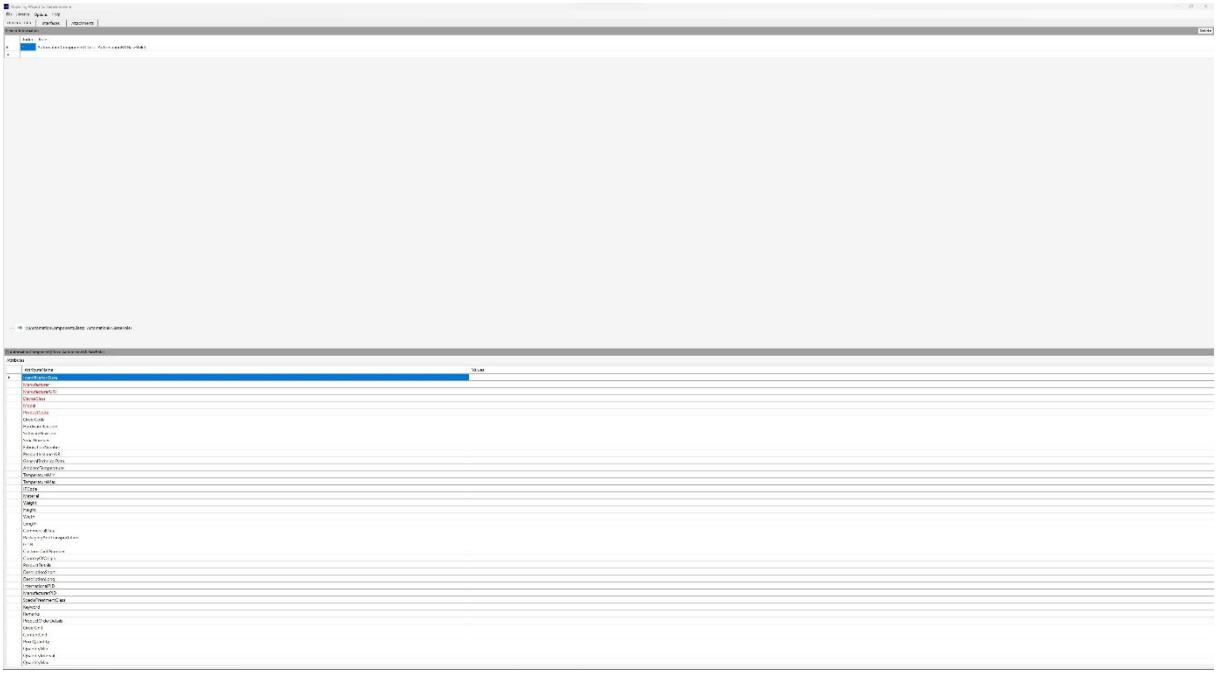
4.4.5 /NF15/Compatibility

The application should be executable on every current system such as Windows 10 or higher. Furthermore, the application is only executable on the Windows platform.

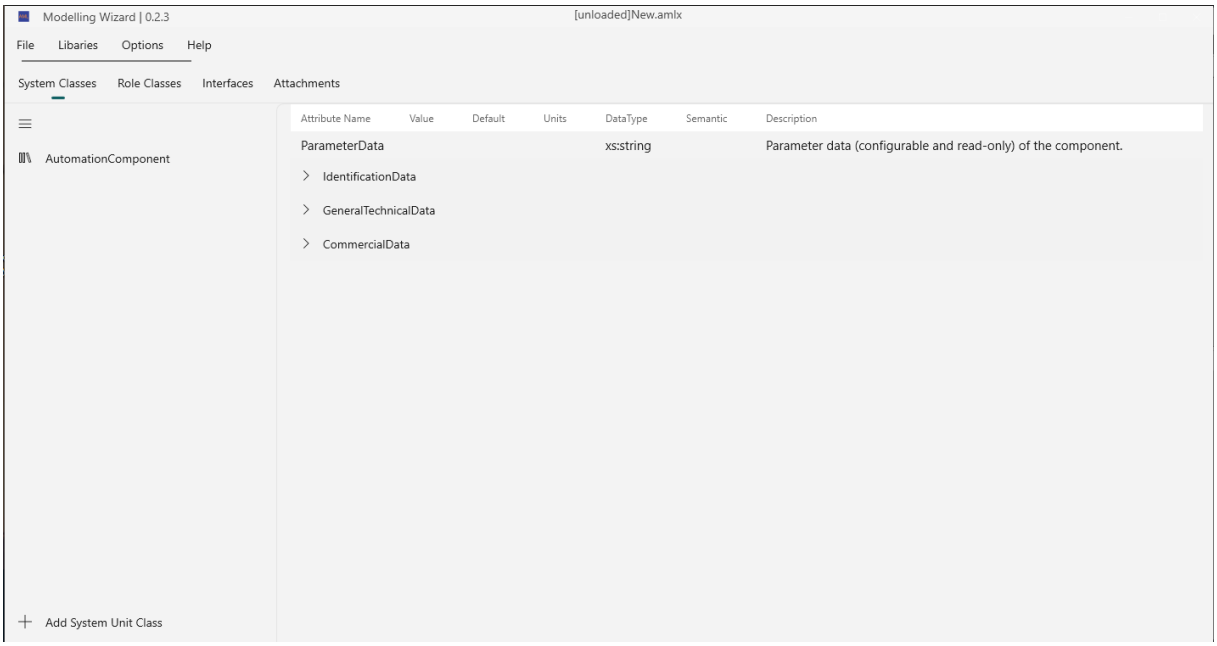
5. Prototypes of the UI

The first prototype includes the changes in the existing UI. The final prototype is based on a completely redesigned interface.

5.1 First Prototype



5.2 Final Prototype



6. References

[1] GitHub Repository from TINF20C