****

**System Requirements Specification**

(TINF21C, SWE)

**Project:** Modelling Wizard Improvements

**Customer:** Markus Rentschler

Christian Holder

**Team:** Project Manager – Robin Ziegler ([inf21100@lehre.dhbw-stuttgart.de](mailto:inf21100@lehre.dhbw-stuttgart.de" \t "_blank))

Developer – Nils Hoffmann ([inf21194@lehre.dhbw-stuttgart.de](mailto:inf21194@lehre.dhbw-stuttgart.de))

Test Manager – Michael Grote ([inf21111@lehre.dhbw-stuttgart.de](mailto:inf21111@lehre.dhbw-stuttgart.de))

System Architect – Fabian Kreuzer ([inf21106@lehre.dhbw-stuttgart.de](mailto:inf21106@lehre.dhbw-stuttgart.de))

Tech. Documentation – Dana Frey ([inf21099@lehre.dhbw-stuttgart.de](mailto:inf21099@lehre.dhbw-stuttgart.de))

Product Manager – Maximilian Trumpp ([inf21123@lehre.dhbw-stuttgart.de](mailto:inf21123@lehre.dhbw-stuttgart.de))

**Change History**

|  |  |  |  |
| --- | --- | --- | --- |
| **Version** | **Date** | **Author** | **Comment** |
| 0.1 | 07.10.2022 | Dana Frey | created |
| 0.2 | 14.10.2022 | Dana Frey | Added ‘Introduction’ and ‘Use Cases’ |
| 0.3 | 17.10.2022 | Dana Frey | Added ‚Features‘ |
| 0.4 | 21.10.2022 | Dana Frey | reworks |
| 0.5 | 23.10.2022 | Dana Frey | Added ‚Enhancements‘ and did reworks |

Table of Contents

[1. Introduction 4](#_Toc117441391)

[1.1 Product Environment 4](#_Toc117441392)

[2. Use Cases 5](#_Toc117441393)

[2.1 <UC.001> Create new device 5](#_Toc117441394)

[2.2 <UC.002> Save device 6](#_Toc117441395)

[2.3 <UC.003> Create interface or load interface from library 6](#_Toc117441396)

[2.4 <UC.004> View device data and device interface data 7](#_Toc117441397)

[2.5 <UC.005> Add attachments to device 8](#_Toc117441398)

[2.6 <UC.006> Format output as CAEX version 2.15/3.0 9](#_Toc117441399)

[3. Features 10](#_Toc117441400)

[3.1.1 <LF11> Import 10](#_Toc117441401)

[3.1.2 <LF12> File validation 10](#_Toc117441402)

[3.1.3 <LF13> Error handling 10](#_Toc117441403)

[3.1.4 <LF14> GUI 11](#_Toc117441404)

[3.1.5 <LF15> Display device in a readable way 11](#_Toc117441405)

[3.1.6 <LF16> Edit device 11](#_Toc117441406)

[3.1.7 <LF17> Create device 11](#_Toc117441407)

[3.1.8 <LF18> Export device 12](#_Toc117441408)

[3.1.9 <LF19> Easy Mode 12](#_Toc117441409)

[3.2.0 <LF20> Advanced Mode 12](#_Toc117441410)

[4. Product data 12](#_Toc117441411)

[5. Bug fixes 12](#_Toc117441412)

[6. Enhancements 12](#_Toc117441413)

[6.1 <ENH10 > Data Models 13](#_Toc117441414)

[6.2 <ENH20 > Device Name 13](#_Toc117441415)

[6.3 <ENH30 > Easy Mode 13](#_Toc117441416)

[6.4 <ENH40 > Error Handling 13](#_Toc117441417)

[6.5 <ENH50 > Remove Input Fields 13](#_Toc117441418)

[6.6 <ENH60 > Save Buttons 13](#_Toc117441419)

[6.7 <ENH70 > Open and Import 13](#_Toc117441420)

[7. References 14](#_Toc117441421)

[8. Glossary 14](#_Toc117441422)

# 1. Introduction

Our project deals with the already existing program called „Modelling Wizard“. The program can create a device model in a graphical user interface (GUI). It is possible to add device interfaces (such as physical ports) or file attachments to the created device model. If the user already owns descriptive files, those files can be used to create a device model as well. The current version of the Modelling Wizard is capable of handling AMLX, AML, EDZ, IODD and GSD imports and CAEX version 2.15/3.0 exports.

The goal of our project is to create a more user-friendly version of the Modelling Wizard which includes an improved version of the GUI as well as fixing bugs and refactoring the code by the team beforehand.

## 1.1 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:

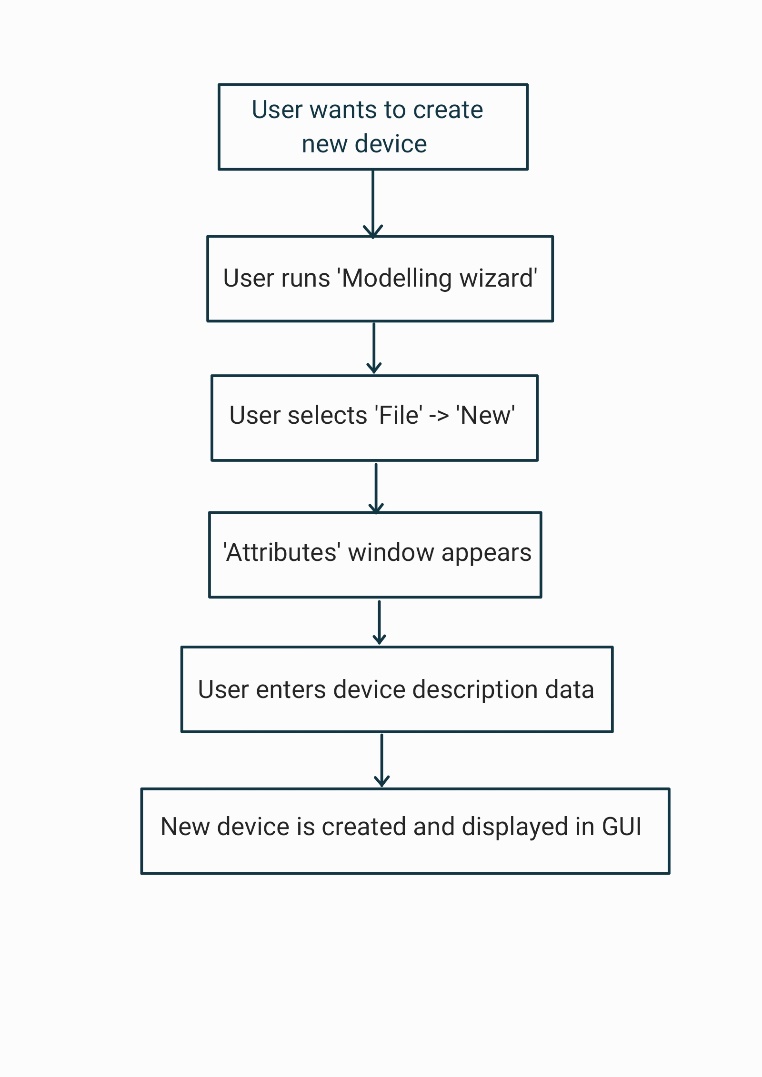
1. CAEX (Computer Aided Engineering Exchange) to describe attributes of objects and their relations in a hierarchical structure. This is called a system topology. In this respect, CAEX forms the overarching integration framework of AutomationML.
2. COLLADA to describe the geometry and 3D Models of an 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.

# 2. Use Cases

Our project doesn't include new added Use Cases and focuses more on the improvement of the given project. Therefore, the following Use Cases have been taken over by the previous team to still be able to visualize the current program’s Use Cases.

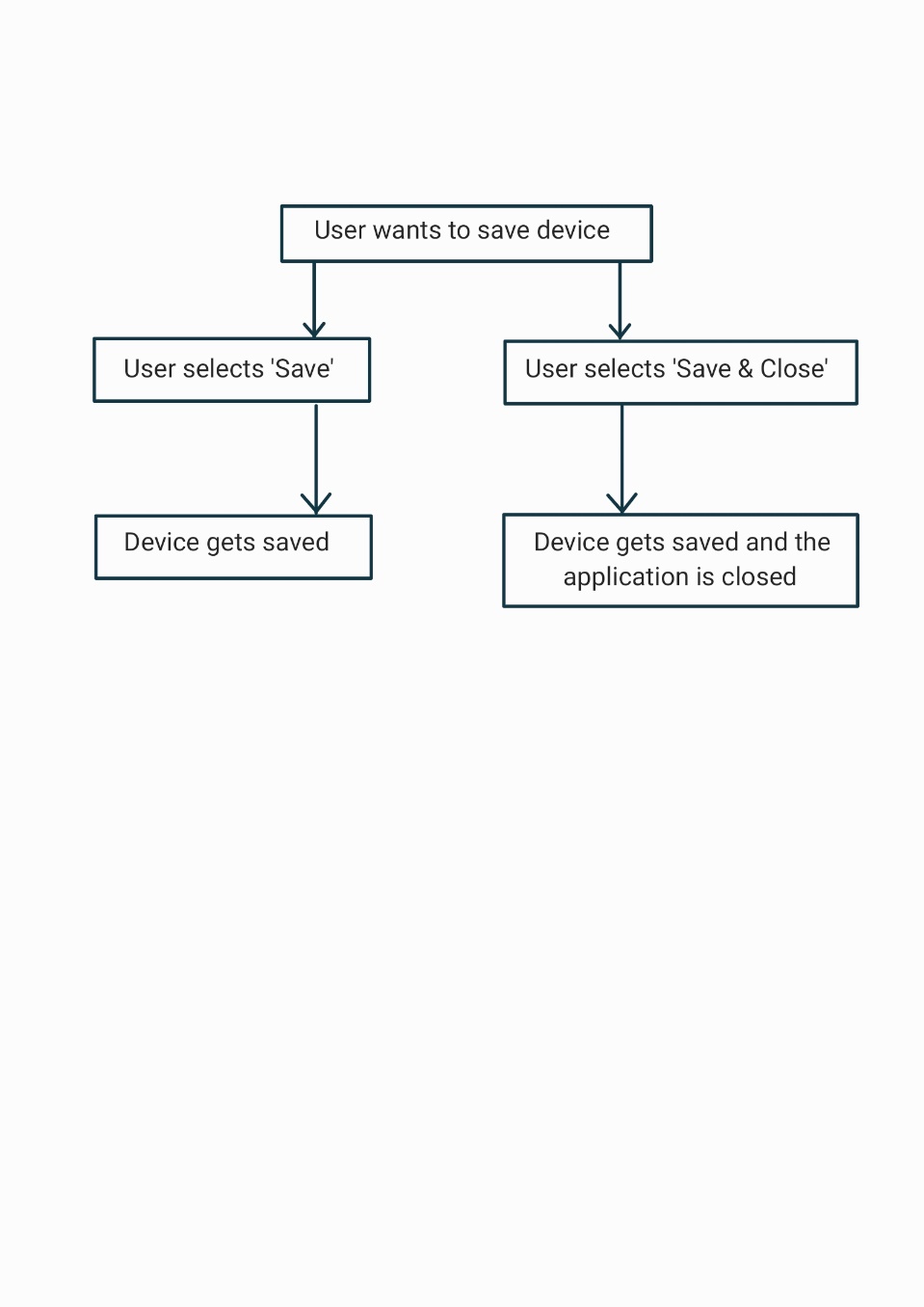
## 2.1 <UC.001> Create new device

|  |  |
| --- | --- |
| Use Case's 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 have the minimal required data for the device on hand. The program needs to be installed on the user’s system and opened. |
| Postcondition on success: | The entered data is displayed completely and correctly |
| Involved Users: | Every end-user of the application |
| Triggering Event: | When the user opens the application and uses the 'New' function to create a new device |



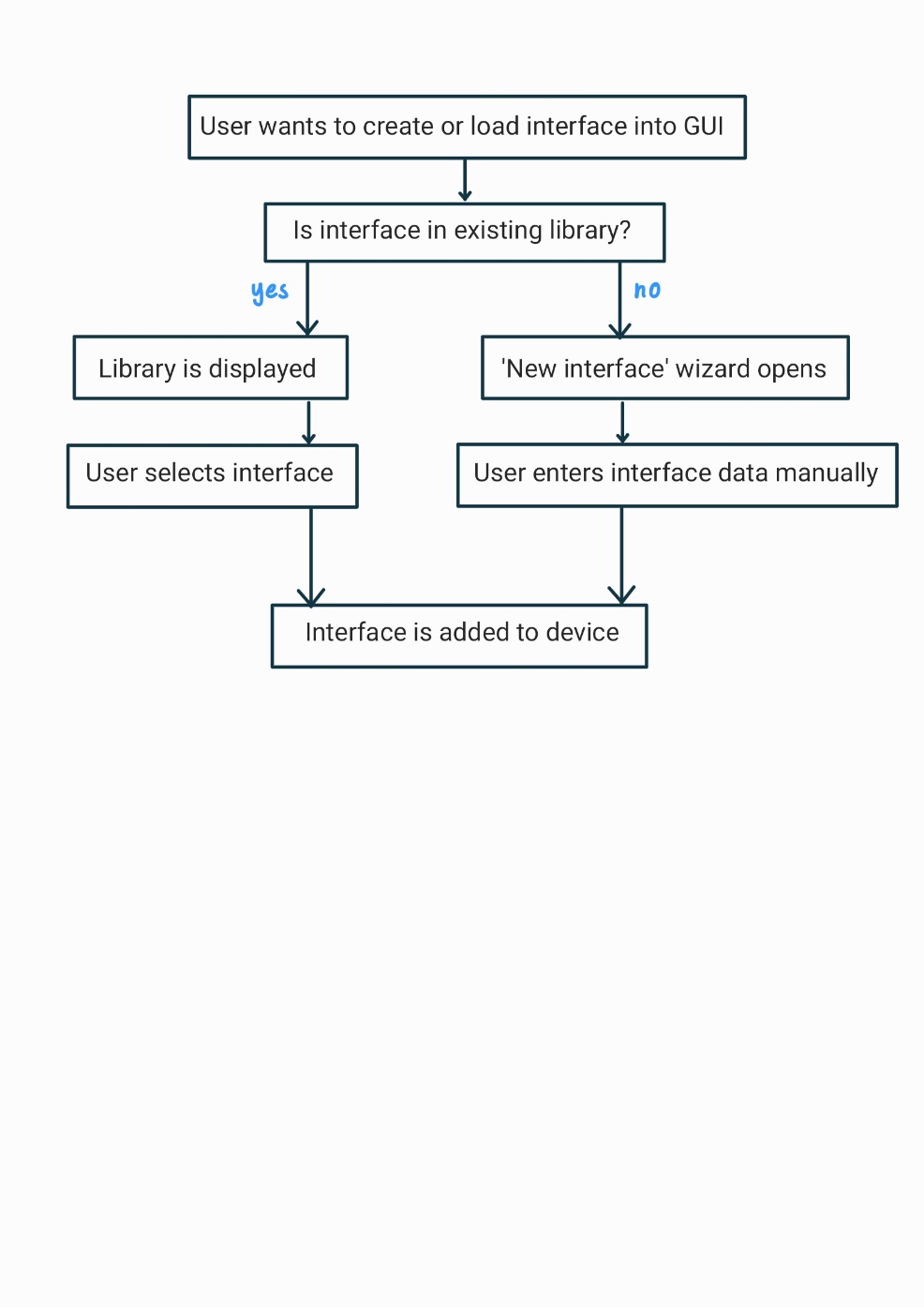
## 2.2 <UC.002> Save device

|  |  |
| --- | --- |
| Use Case's Objective: | User wants to save a device by either using the ‘Save’ or the ‘Save & Close’ function |
| System Boundary: | The application itself |
| Precondition: | The user created a device |
| Postcondition on success: | The user created or loaded at least one device successfully |
| Involved Users: | Every end-user of the application |
| Triggering Event: | When the user has an opened device which the user wants to save |



## 2.3 <UC.003> Create interface or load interface from library

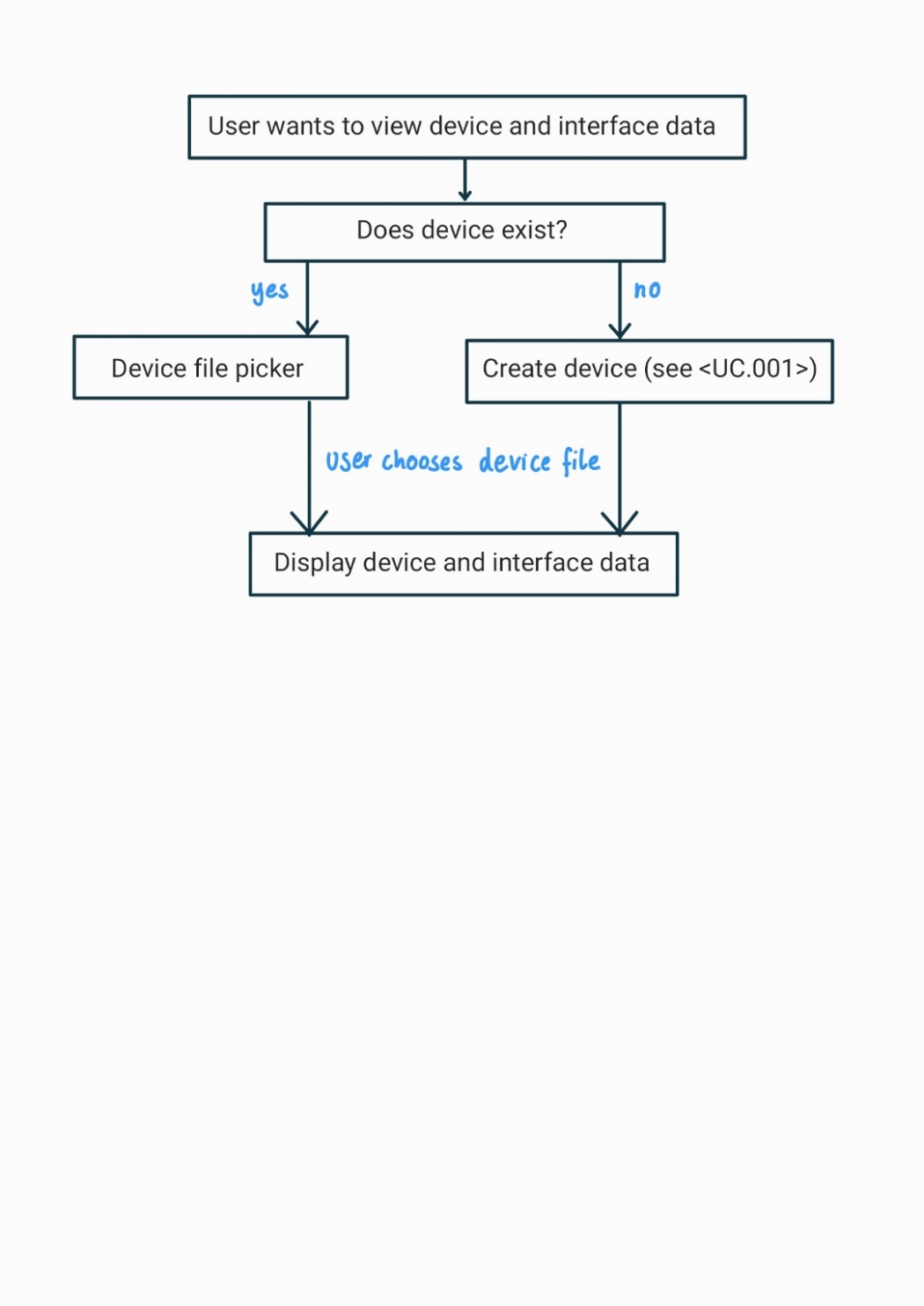
|  |  |
| --- | --- |
| Use Case's Objective: | Creating a device interface by inserting the data manually into the user interface. Or to add an interface from one of the existing libraries |
| System Boundary: | The application itself |
| Precondition: | The user needs to have the minimal required data for the device or interface to be added. |
| Postcondition on success: | The user has submitted the specific data completely and correctly |
| Involved Users: | Every end-user of the application |
| Triggering Event: | When the user wants to add or create a device interface |



Supported input formats are: AMLX, AML, EDZ, IODD, GSD

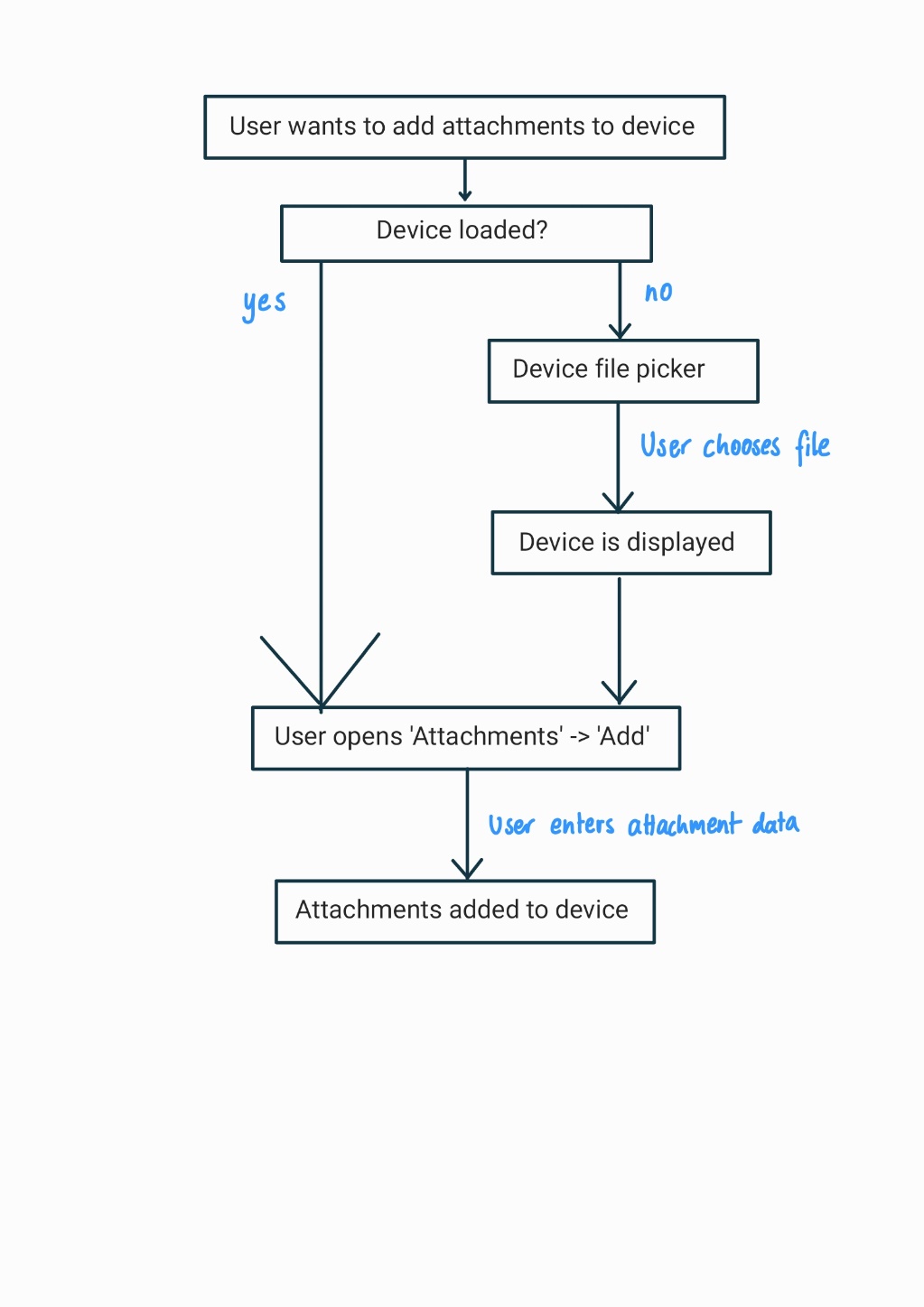
## 2.4 <UC.004> View device data and device interface data

|  |  |
| --- | --- |
| Use Case's Objective: | After at least one device was successfully created, the device data should be visible and editable on the user interface |
| System Boundary: | The application itself |
| Precondition: | The user created or loaded a device |
| Postcondition on success: | The user created or loaded at least one device successfully |
| Involved Users: | Every end-user of the application |
| Triggering Event: | When the user wants to view device data and device interface data |



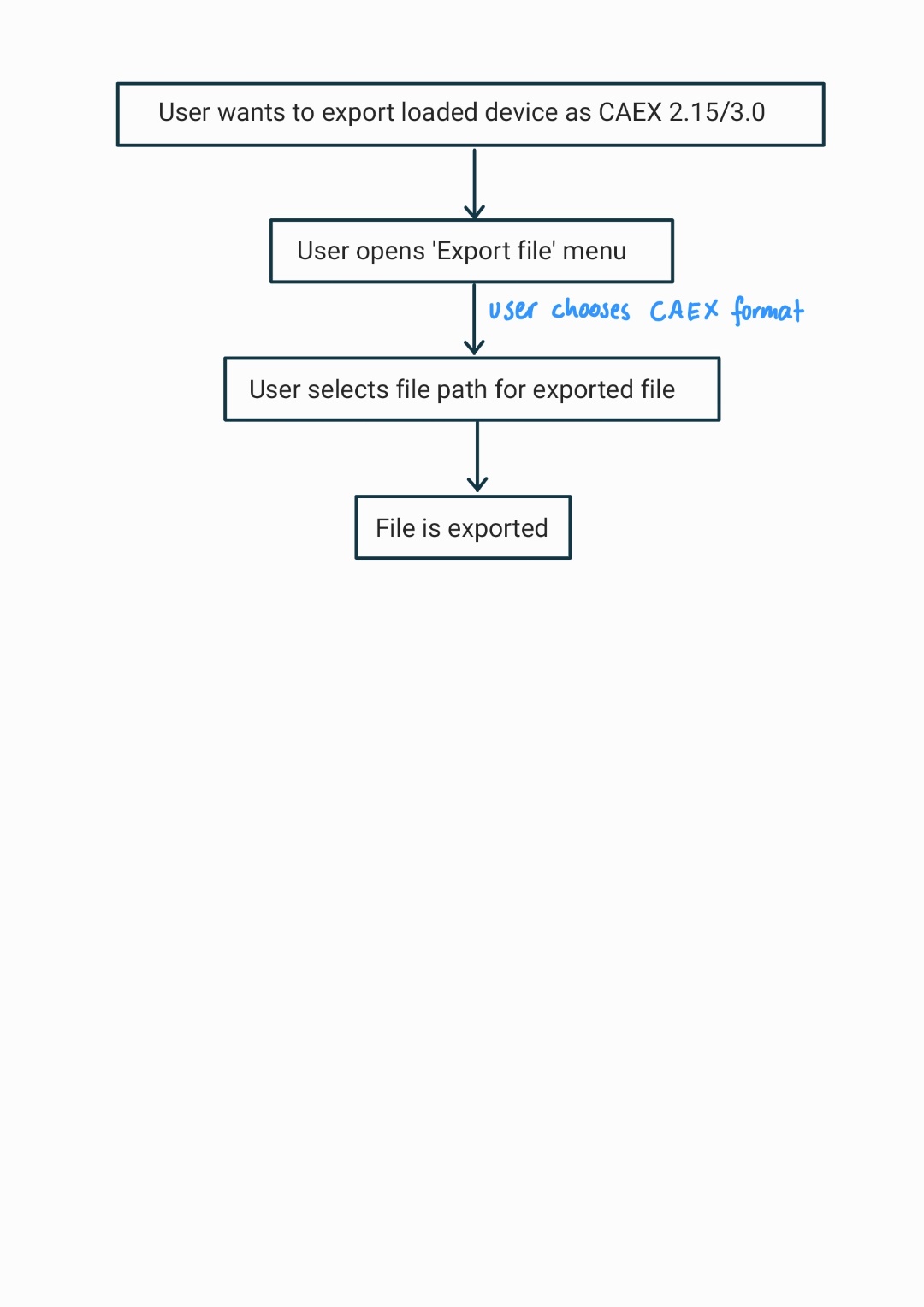
## 2.5 <UC.005> Add attachments to device

|  |  |
| --- | --- |
| Use Case's Objective: | It is possible to add an attachment to the object, such as a manufacturer's icon |
| System Boundary: | The application itself |
| Precondition: | The user created or loaded a device |
| Postcondition on success: | The user created or loaded at least one device successfully |
| Involved Users: | Every end-user of the application |
| Triggering Event: | When the user has the need to edit device data and add attachments such as icons. |



## 2.6 <UC.006> Format output as CAEX version 2.15/3.0

|  |  |
| --- | --- |
| Use Case's Objective: | Make the export to CAEX formats possible for devices |
| System Boundary: | The application itself |
| Precondition: | The user created or loaded a device |
| Postcondition on success: | The user created or loaded at least one device successfully |
| Involved Users: | Every end-user of the application |
| Triggering Event: | When the user wants to save a device in the CAEX format |



# 3. Features

## 3.1.1 <LF11> Import

The application should be able to import a file by the absolute path to the file. This import supports files of the file types AMLX, AML, EDZ, IODD and GSD.

## 3.1.2 <LF12> File validation

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

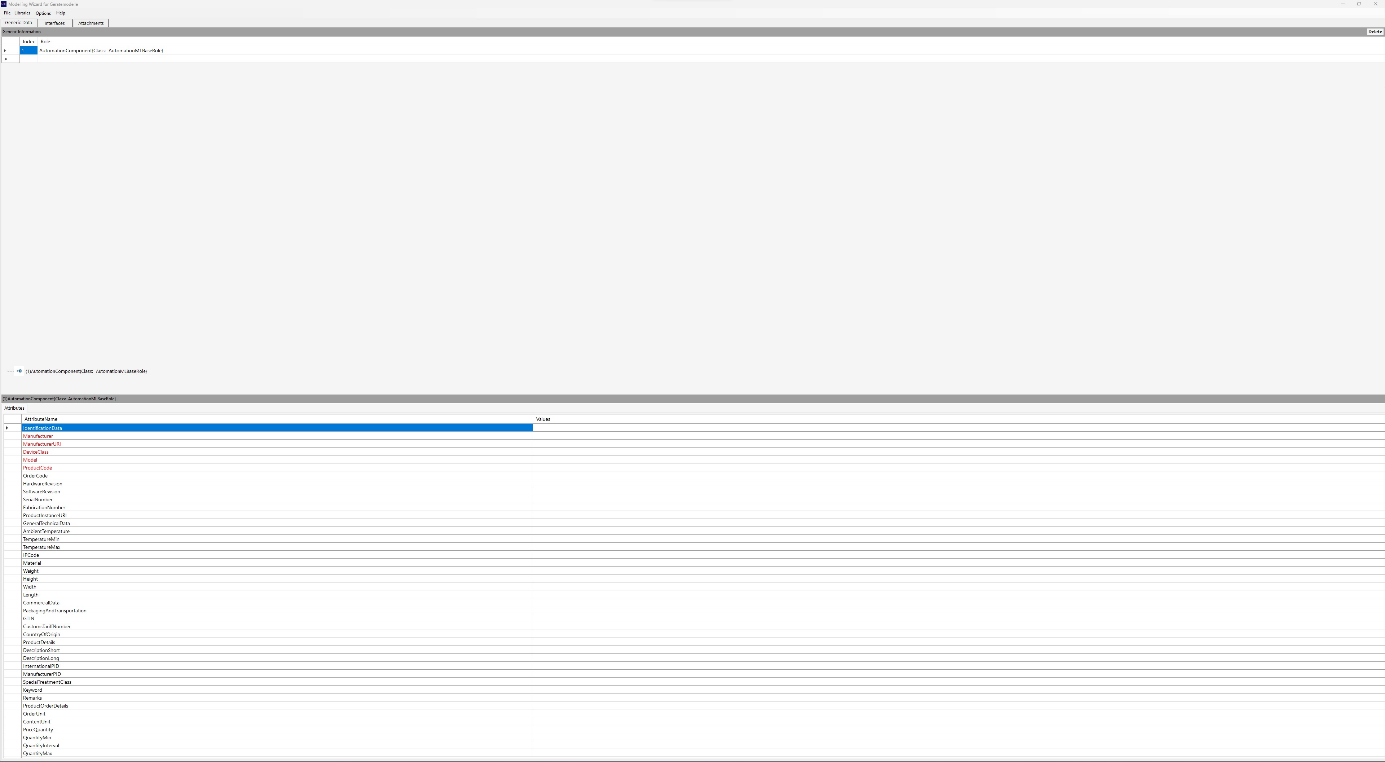
## 3.1.3 <LF13> Error handling

The system shall be able to handle errors (unexpected shut down, wrongly formatted files, ...) and throw an error to the user. If an error appears to be in a specific part of the program, the cursor should be pointed to that exact location.

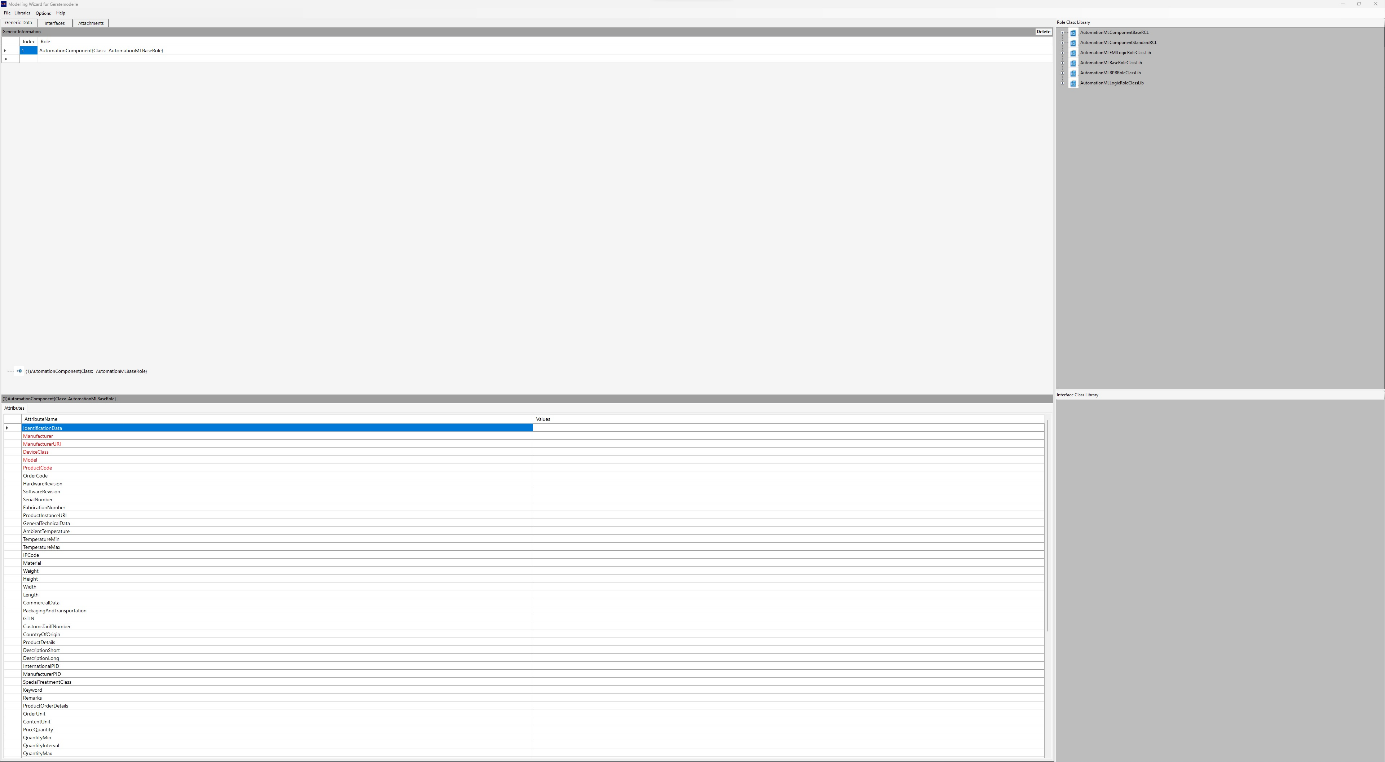
## 3.1.4 <LF14> GUI

The system should display a graphical user interface after startup of the standalone application. The user will interact with this GUI for every other functionality of the application. For this, the application must be converted from a plugin to a full application on its own.

GUI in easy mode:



GUI in advanced mode:



## 3.1.5 <LF15> Display device in a readable way

When a device is loaded or created the attributes of the element should be displayed directly and easily readable for the user.

## 3.1.6 <LF16> Edit device

When the attributes of a loaded device are displayed to the user, the user should be able to edit every attribute he wants to change.

## 3.1.7 <LF17> Create device

When the application is started, the user should be able to create a new and empty device model.

## 3.1.8 <LF18> Export device

When the user has edited a device, he should be able to save the device to a file.

## 3.1.9 <LF19> Easy Mode

## 3.2.0 <LF20> Advanced Mode

# 4. Product data

# 5. Bug fixes

*Yet to be determined.*

# 6. Enhancements

## 6.1 <ENH10 > Data Models

The drag and drop Feature should be removed and replaced with options for opening a file. At the start of the program, all the data models should be open and displaying all the information at once.

## 6.2 <ENH20 > Device Name

The product code should replace the device name.

## 6.3 <ENH30 > Easy Mode

The data models on the right side of the program should be removed. The focus should lay on the inputs at the bottom.

## 6.4 <ENH40 > Error Handling

The program should deliver an active input for where the failure was detected.

## 6.5 <ENH50 > Remove Input Fields

The input fields for ‘name’ and ‘producer’ should be removed and the inputs from the bottom should be used.

## 6.6 <ENH60 > Save Buttons

The buttons ‘Save’ and ‘Save & Close’ should be consolidated into one button.

## 6.7 <ENH70 > Open and Import

The buttons ‘Open’ and ‘Import’ should be consolidated into one button.

# 7. References

[1] <https://github.com/H4CK3R-01/TINF20C_ModellingWizard_Devices/wiki/1.-Software-Requirements--Specification#UC3>

# 8. Glossary

|  |  |
| --- | --- |
| 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 |
| CAEX | Computer-Aided Engineering Exchange |
| EDZ | EPLAN Electric P8 Data Archive Zipped File |
| GSD | General Station Description |
| GUI | Graphical User Interface |
| IODD | Input/Output Device Description |