DHBW
Duale Hochschule
Baden-Württemberg
Stuttgart

System Architecture 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)

Change History

Version	Date	Author	Comment
0.1	13.10.2022	Fabian Kreuzer	created
0.2	16.10.2022	Nils Hoffmann	Added points 1-3
0.3	18.10.2022	Nils Hoffmann	Rework of some chapters
1.0	20.10.2022	Fabian Kreuzer	Visual updates

Table of Contents

Introduction

This project targets the improvement and debugging of an existing standalone application which was based on a plugin for the AutomationML editor. This includes modifications on the existing "Easy/Advanced mode" functionality and the "Import" option. Also, improvements on the general usability and user-friendliness are intended in the events of this project. Furthermore, there are multiple cases where it is planned to fix bugs and errors

Glossar

AML	*Automation mark-up Language is an open standard data format for storing and exchanging plant planning data.*
AMLX	AML Package to store also not AML files in one package.
CAX	File format of AML Device files.
C#	High level language often used for programming
GUI	*Graphical User Interface*
.NET	*The .NET Framework is a software development and runtime environment developed by Microsoft for Microsoft Windows.*
AML DD	AML Device Description
GSD	General Station Description, data format for Profibus and Profinet devices.
IODD	*IO Device Description describes the sensors and other participants in an IO-Link network.*

System Overview

The system is currently a standalone application for windows, with the problem that it is hard to use, and it needs some time to get familiar with the tool. In addition to that

some of the functions aren’t working properly or aren’t necessary for basic Use-cases.

System Environment

The standalone application can only be accessed via Windows, as this operating system is used as the platform. The application can be installed and the graphical user interface can be used via this platform.

Software Environment

For the standalone application to work, you need at least version 4.5 of the .Net framework, as C# was used developing it. This version of the framework is available from the "Windows Vista" operating system variant onwards. The application can be started either by using the finished .exe application or running the source code in Microsoft Visual Studio.

Quality Goals

In order to achieve the quality goals, different criteria are considered.

Runtime Quality Attributs

These can be observed at execution time of the system.

Usability

Usability is one important aspect of this project. The application should be easy and intuitive to use. The user should get to the point where he wants to be in the easiest way possible to allow fast and result-orientated working with the application. This includes grouping and fusing similar areas together like for example the “Import” and “Load” options. At this moment there are some areas in the user interface that are unnecessary and confusing to basic users and are counteracting against the intuitiveness.

Functionality

The application should enable any user to utilize it with every function that it offers. Bugs and errors are preventing this and stop the user from efficient working. Furthermore when an error appears the user should get closer information on the problem in order to perform steps to solve the issue and use the intended function. Also it is planned to implement further functions to improve the functionality even more.

Availability

The programme and its code will be available on GitHub on a public repository. This means that anyone can access the programme at any time.

Interoperability

This is important because it must be ensured that users are not firmly bound to this standard or programme. After all, the main users will come from the industry and therefore place a lot of value on a uniform standard. This includes being able to export and import common datatypes.

Non-Runtime quality attributes

These cannnot be observed at execution time of the system.

Modifiability

Since the programme is open source and publicly viewable, it can be extended by anyone. This is important if any use cases or issues arise in the future.

Portability and reusability

To ensure portability and reusability it is necessary to create libraries and documentation files. These can help future Users to understand and enhance the application.

Quality Concept

This part of the software architecture specification explains and breaks down the problems that usually arise during the further development of software. This includes concepts for dealing with these problems and thus improving the quality of the final product.

Usability Concept

The criteria for good usability are:

Intuitiveness: Intuitiveness is a key component of good usability which means that the user understands the application without a long training period. This point is also interesting for companies, as they then do not have to train their employees for so long. This is to be achieved by having important things at the top and not hidden in a drop-down selection. Furthermore, the design and layout must be adopted from other well-known programmes. Also as already earlier said it is important that similar functions and areas are grouped and maybe even fused together to ease the use even more.

Design: When designing the application, an appealing layout is important. It should be clearly recognisable which function is hidden behind which visual elements and how the user can navigate through the app.

Recognition value: This means that similar functions should be realized with the same sequences. This makes it easier for the user to find his way around the functions and increases user friendliness. For example, when creating models or a wizard that guides you through the creation.

Colour scheme: A good colouration can be created by an attractive choice of colours and their coordinated contrasts. At the moment, matte colours are more in vogue and are preferred for designs. Many people find these more pleasant.

Implementation guideline:

After conducting the usability test, guidelines were developed based on which of the GUI improvements will be made. Therefore, the GUI must meet the following guidelines:

- A consistent colour palette should be used, where elements with the same events have the same colours for recognition. Matte and colour-coordinated palettes should be used here.
- Contrasts, borders as well as roundings should be used to emphasise inputs or interaction fields. Blank areas should be created to give an attractive and uncluttered design. Another advantage is that the user will then find his way around more easily.
- The design and layout should be self-explanatory and reinforce the previous point to allow intuitive use.
- For the easy mode unnecessary information should be hidden to create a more clear and simple overview.

Based on the criteria just defined and the guidelines developed, the graphic interface is adapted and optimised. Functionality is very important but should not negatively affect usability. Nevertheless, compromises have to be made in terms of feasibility, as the basic concept of AutomationML should be presupposed.

Code Quality

Code quality is one of the most important aspects when it comes to software that is being developed further and may come from different developers. For this reason, we have made it our mission to address and improve the problem of code quality. To maintain a certain standard, we have agreed on certain conventions:

- Commenting on sections of code that are not clearly understandable in order to explain the implemented idea to others
- Programming paradigms and programming principles that make the code easy to understand, such as one main function per file and a sensible folder structure.
- In general the source code should be compliant with the common “Clean Code” guidelines

These lead to the point that the code is indented uniformly to improve readability and comprehension. Furthermore, the program is divided into modules that can then be imported and used. Otherwise, you run the risk of having to write duplicate code. In addition, it is our responsibility to write documentation that records which ideas have been implemented and how, so that the existing functions are easier to understand for future developments and can be built upon.

Architectural Concept

The system architecture is based on the previous Project, made by Students of the DHBW Stuttgart. The Main Goal is to update the GUI and bugfix the Application. Therefore, there will only be slight changes in the Software architecture.

Architectural Model

The application is designed according to a Model-View-Control (MVC) architecture pattern, which could be best described as a cycle. In this architecture the user can only see and interact with the application via the GUI. But the actual processing is not done in the GUI the background processing of the input is done by the controller. The controller then interacts with the model which updates the view (cf. Figure 1).

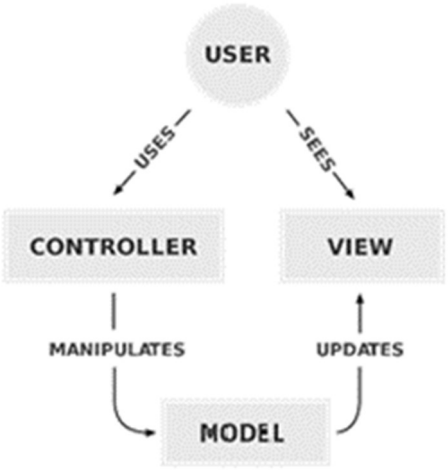


Figure 1 – MVC Architecture

Almost all the logic is contained in the controller, which thus forms the centre of the entire system architecture and contains the functionalities. Therefore only one layer is accessible to the user, the GUI.

Component Diagram

The controller is the main control unit from the plugin. It is responsible for communicating with the user interface and the external systems, which were added for conversions. This interface is the heart of the entire application and is responsible for all the functionalities, but also for the integration of additional functions like saving or loading AMLX packages from the AutomationML Engine. Thus, the concept builds on that of the plugin, making it easier to adapt functions and ideas. These ideas are mostly working poorly or not working at all. The work to do is to test these functions and fix the problems.

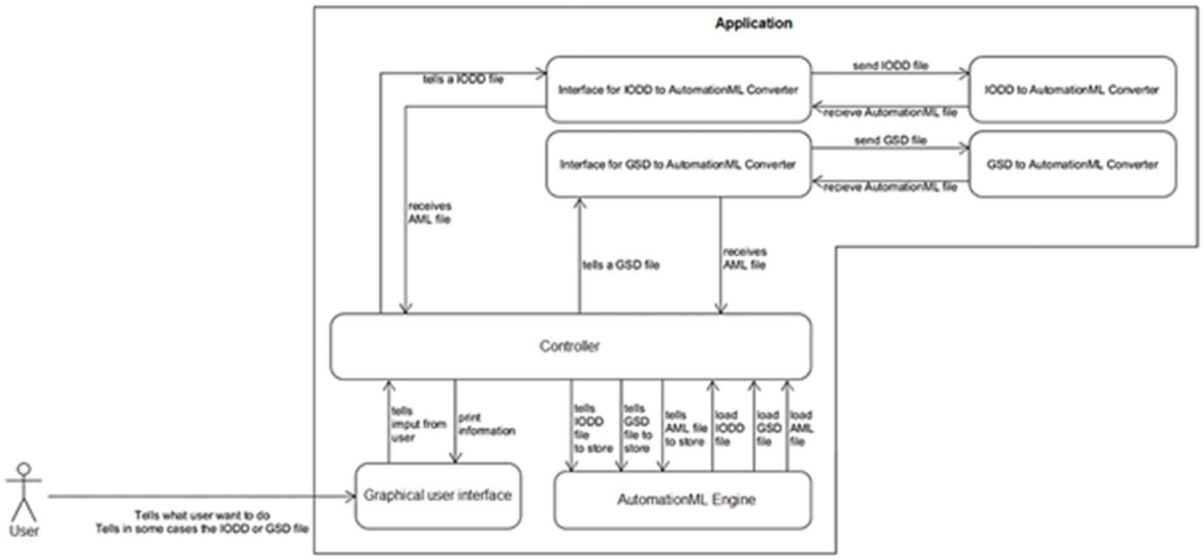


Figure 2 - Logic of the plugin

In the figure above, you can see the new architecture design that depicts the structure of the application (cf. Figure 2). As you can see in the illustration, the user only interacts with the graphical user interface. This then passes the input from the user to the controller and the controller displays the information in the GUI. The controller processes the requests with the help of the AutomationML engine, but not all functions were mapped for this because there would be too many. For example, it can be used to save and load models in AML format. To be able to process IODD and GSD model formats, the programme needs converters. These work with two interfaces and return an AutomationML file. However, due to the further development based on the project, the architecture became more and more unstructured and complex. As a result, MVC is no longer used as intended. This was further complicated by the use of a Microsoft Forms application. Thus the former GUI is now mapped on a Windows Forms object.

System design

The program is designed as a stand-alone application with input from the Modelling wizard. The “Program” class receives information from the Modelling wizard plugin and sends it to the GUI called “Form1” which displays the information for the user to interact with.

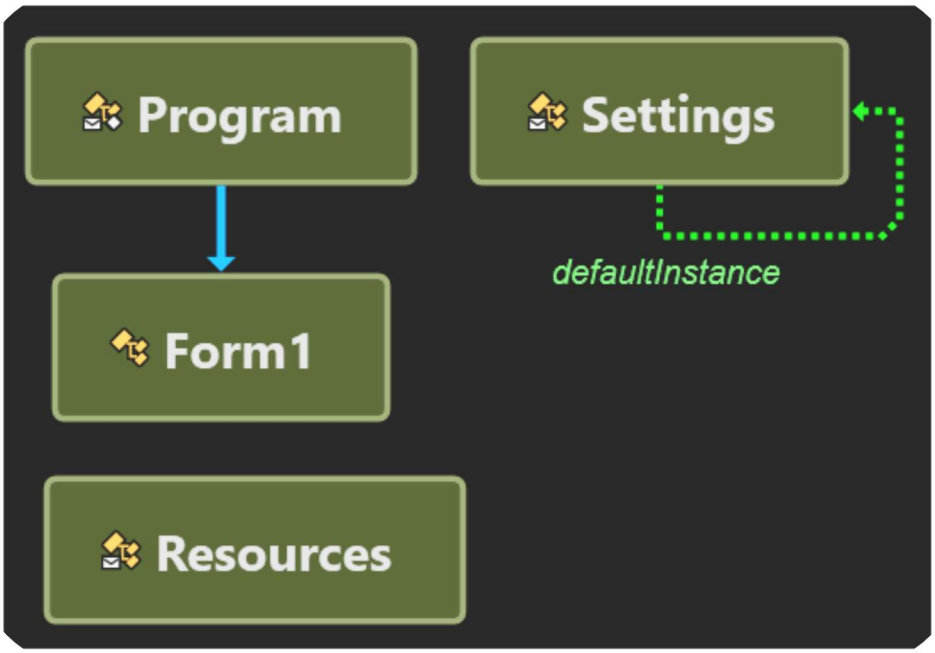
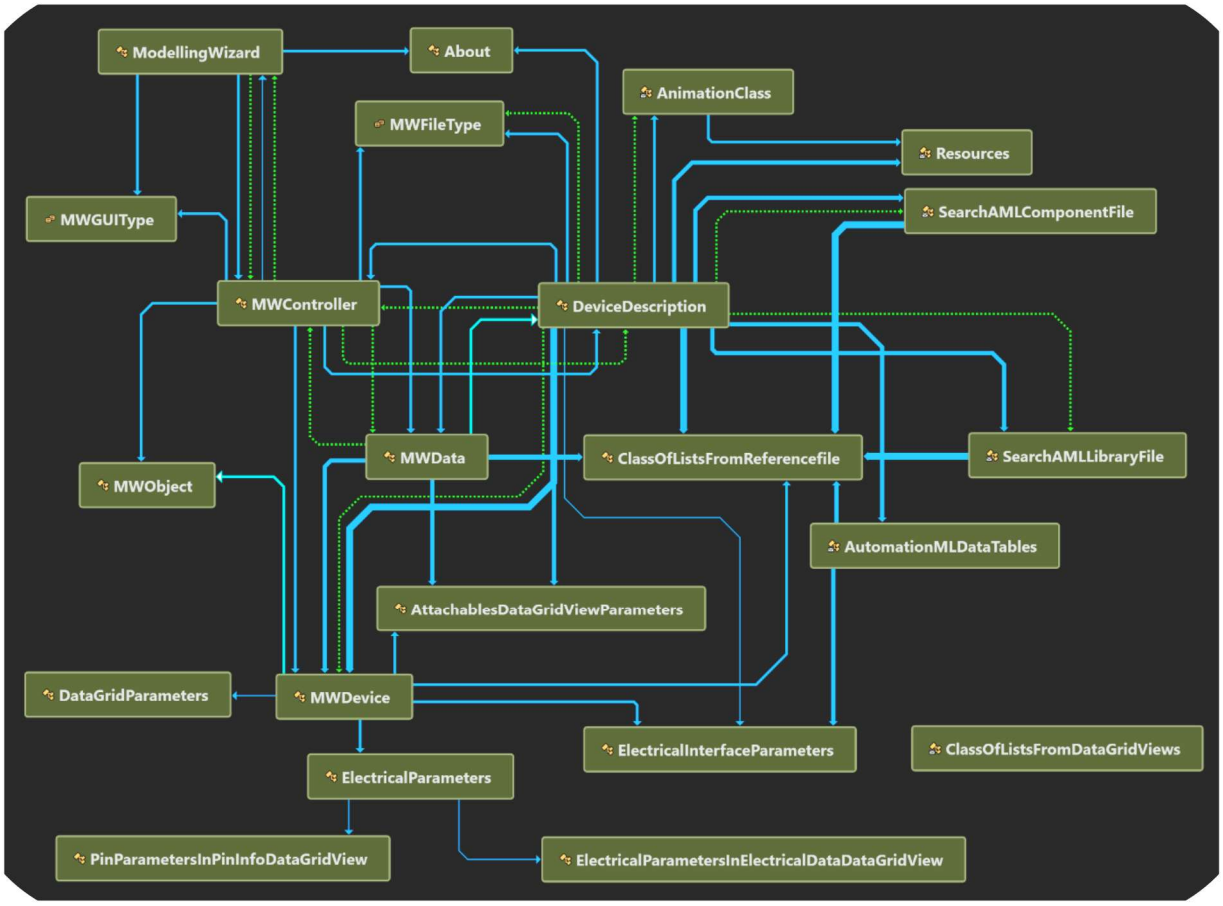
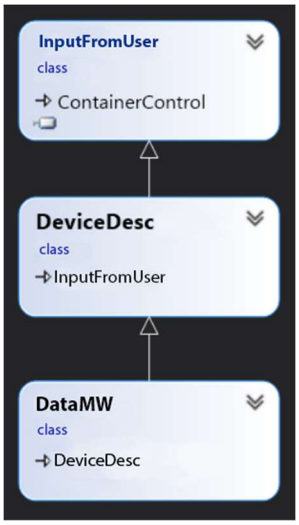


Figure 3 - Class design from the main application

The "Program" field in the diagram is only loading the Modelling wizard plugin and puts the information in "Form1". This function then displays the "ModellingWizard", i.e. the plugin.

Figure 4 - Class design from the Modelling Wizard





The current design has not changed to the predecessor project (cf. Figure 5). The concept of the plugin and the code was carried over from the old project.

Still the MVC pattern is a small part of the whole system design (cf. Figure 6). The GUI backend and errorhandeling for user Input is handeld by a separate Controller DeviceDesc, DataMW generates an Data Object witch is displayed in the GUI.

Figure 5 - MVC pattern

Classname	*Storage location*
About	https://github.com/robinziegler/TINF21C_Team4_Modelling_Wizard_Improvements/tree/master/SOURCE/Plugin/About.xaml.cs
AnimationClass	https://github.com/robinziegler/TINF21C_Team4_Modelling_Wizard_Improvements/tree/master/SOURCE/Plugin/AnimationClass.cs
AutomationMLDataTables	https://github.com/robinziegler/TINF21C_Team4_Modelling_Wizard_Improvements/tree/master/SOURCE/Plugin/AutomationMLDataTables.cs
ClassOfListFromReferenceFile	https://github.com/robinziegler/TINF21C_Team4_Modelling_Wizard_Improvements/tree/master/SOURCE/Plugin/ClassOfListsFromReferencefile.cs
DeviceDescription	GUI: https://github.com/robinziegler/TINF21C_Team4_Modelling_Wizard_Improvements/tree/master/SOURCE/Plugin/DeviceDescription.Designer.cs *Logic:* https://github.com/robinziegler/TINF21C_Team4_Modelling_Wizard_Improvements/tree/master/SOURCE/Plugin/DeviceDescription.cs
ModellingWizard	https://github.com/robinziegler/TINF21C_Team4_Modelling_Wizard_Improvements/tree/master/SOURCE/Plugin/ModellingWizard.xaml.cs
MWController	https://github.com/robinziegler/TINF21C_Team4_Modelling_Wizard_Improvements/tree/master/SOURCE/Plugin/MWController.cs
MWData	https://github.com/robinziegler/TINF21C_Team4_Modelling_Wizard_Improvements/tree/master/SOURCE/Plugin/MWData.cs
MWDevice	https://github.com/robinziegler/TINF21C_Team4_Modelling_Wizard_Improvements/tree/master/SOURCE/Plugin/MWDevice.cs
Resources	https://github.com/robinziegler/TINF21C_Team4_Modelling_Wizard_Improvements/tree/master/SOURCE/Plugin/Resources
SearchAMLComponentFile	https://github.com/robinziegler/TINF21C_Team4_Modelling_Wizard_Improvements/tree/master/SOURCE/Plugin/SearchAMLComponentFile.cs
SearchAMLLibraryFile	https://github.com/robinziegler/TINF21C_Team4_Modelling_Wizard_Improvements/tree/master/SOURCE/Plugin/SearchAMLLibraryFile.cs

Subsystem specification

All modules have no external data.

<MOD.001>: Graphical User Interface (GUI)

<MOD.001>	*Graphical User Interface*
System requirements covered:	*/LF10/, /LF30/, /LF40/, /LF50/, /LF60/, /LF70/, /LF80/, /LD20/*
Service:	*The graphical user interface is taking input from the user and sending it to the controller by calling event functions.*
Interfaces:	---
Storage location:	https://github.com/robinziegler/TINF21C_Team4_Modelling_Wizard_Improvements/tree/master/SOURCE/Plugin/DeviceDescription.Designer.cs
Modul documentation:	https://github.com/H4CK3R-01/TINF20C_ModellingWizard_Devices/wiki/MOD.001:-Graphical-User-Interface-(GUI)

<MOD.002>: Controller

<MOD.002>	*Controller*
System requirements covered:	*/LF10/, /LF20/, /LF30/, /LF60/, /LF70/, /LF80/, /LD10/, /LD20/*
Service:	*Logic distribution is handled by the controller. It is reacting to the events triggered by the GUI and takes care of creating the respective objects. Also the input and output functions are implemented in the controller.*
Interfaces:	*Interface of AMLX packages. For export/import of amlx files there is the MWData class.*
Storage location:	https://github.com/robinziegler/TINF21C_Team4_Modelling_Wizard_Improvements/tree/master/SOURCE/Plugin/DeviceDescription.cs
Modul documentation:	https://github.com/H4CK3R-01/TINF20C_ModellingWizard_Devices/wiki/MOD.002:-Controller

<MOD.003>: Runtime

<MOD.003>:	*Runtime*

System requirements covered:	*/LF40/*
Service:	*This part of the programme turns the plug-in into a standalone application. The goal is that it has the same feature set as the AutomationML editor on which the plugin runs.*
Interfaces:	---
Storage location:	https://github.com/robinziegler/TINF21C_Team4_Modelling_Wizard_Improvements/tree/master/SOURCE/Application/Program.cs
Module documentation	https://github.com/H4CK3R-01/TINF20C_ModellingWizard_Devices/wiki/MOD.003:-Runtime

Technical Concepts

Persistence

The Usage of the publicly available "AMLX" package standard allows the system to be persistent. It also makes the created or edited models available via the AutomationML Editor. Having a persistent application is best used for an industrial environment.

Communication with other IT-Systems

The plugin already had the problem that it was dependent on external programmes. Thus, "IODD" and "GSD" programme types had to be converted in order to be able to use them. currently the conversion is somewhat implemented and or goal is to fix the occurring bugs. The advantage of this is that the user does not have to install external dependencies which might cause even more uncontrollable errors. This means that, in the best case, the only cause of action from the user to load "IODD" and "GSD" file types is by using the Windows file system.

Deployment

To develop the application, one has to install a valid version of Visual Studio, the most recent version is “Visual Studio 2022”. As for it now to use the application it can be installed via a installer, we might have to change the application into a single executable.

Data validation

All data checks are running in the background, invisible for the user. The controller is checking for missing information and incorrect entries, that must be specified as mandatory information for example before a conversion can take place, the input file (GSM, IODD and CSP+) needs to be vali-dated to ensure a conversion is possible. The data validation currently checks for the wrong entries while loading a file. It should be able to at least load a file without every mandatory entry, but not save it.

Exception handling

Exception handling is necessary to prevent errors caused by the user while using the program. So called "try-catch" blocks are used to 'catch' these and prevent unwanted or incorrect behavior of the software. This way, the user is informed about what did not work and he/she may be able to fix the problem. For user compatibility if point 7.4 throws an exeption while saving the user should be shown the exact location where the error is in the GUI.

Figures