System Architecture Specification

(Architekturspezifikation)

(TINF19C, SWE I Praxisprojekt 2020/2021)

Project: *Modelling Wizard*

Customer: *Rentschler & Holder*
Rotebühlplatz 41
70178 Stuttgart

Supplier: by Simon Jess - Team 2
(Simon Jess, Timo Zaoral, Stefan Banov, Tobias Roth, Phillip Tran)
Rotebühlplatz 41
70178 Stuttgart

Version	Date	Author	Comment
0.1	07.09.2020	Simon Jess	Created
0.2	26.10.2020	Simon Jess	Introduction and system overview
0.3	04.11.2020	Simon Jess	Architectural concept
0.4	05.11.2020	Simon Jess	System design, subsystem specification and technical concepts
0.5	09.11.2020	Simon Jess	Finalize after Review
1.0	12.11.2020	Team 2	Completion and formalization

Contents

1. Introduction

The goal of this project is to further develop and improve a plugin for the AutomationML editor. Main part is the improvement of the graphical user interface. To achieve this, the usability is one of the main components. Furthermore, the existing bugs should be handled. The goal is to be able to create, save and edit files in the CAX 2.15 and 3.0 standard.

1.1. Glossar

.NET	The .NET Framework is a software development and runtime environment developed by Microsoft for Microsoft Windows.
C#	High level language often used for programming
GUI	Graphical User Interface
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

2. System Overview

2.1. System Environment

The way to access and work with the plugin is via the AutomationML editor. There you can install the plugin and use the graphical interface. In the plugin you can create and edit AMLX packages to use them in the AutomationML editor.
Among others the IODD and GSD converter are used as neighboring systems.

2.2. Software Environment

That the plugin works you need at least version 4.5 of the .Net framework, because it was developed in C# using the .net framework. This version of the framework is available from Windows Vista or later. Furthermore, the plugin is only available in the AutomationML editor and is not a standalone software.

2.3. Quality Goals

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

2.3.1. Usability

Usability is the key aspect in the whole project. For this purpose, a GUI was created to enable the user to use it as easy as possible. Intuitive control is very important, but also an attractive design is necessary to create the highest possible user experience. High user experience is essential for a plugin to be successful in simplifying work steps. **(further information in the usability concept)**

2.3.2. Bug fixing

Functions that are already implemented should be fixed, that the plugin works without any errors or bugs, which are not intended. Another milestone to keep the quality high is the fixing of bugs that cause unwanted behavior or even fatal errors.

3. Architectural Concept

The Plugin is based on the work of a student team and a master student of the company Balluff, which already programmed the basic functionalities. Main part of the architecture is the GUI and the controller. The GUI displays all functionalities and the controller contains the logic.

3.1. Architectural Model

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

The controller is the main control unit. It is responsible for the communication with the GUI and the external systems added for conversions. This interface is the core of the whole plugin and is responsible for functionalities, but also the integration of additional functions like saving or loading AMLX packages. In the Modelling Wizard project, no additional model was used, but the controller also takes over the functions. More details in the explanation of the Model-View-Control (MVC) architecture principle.

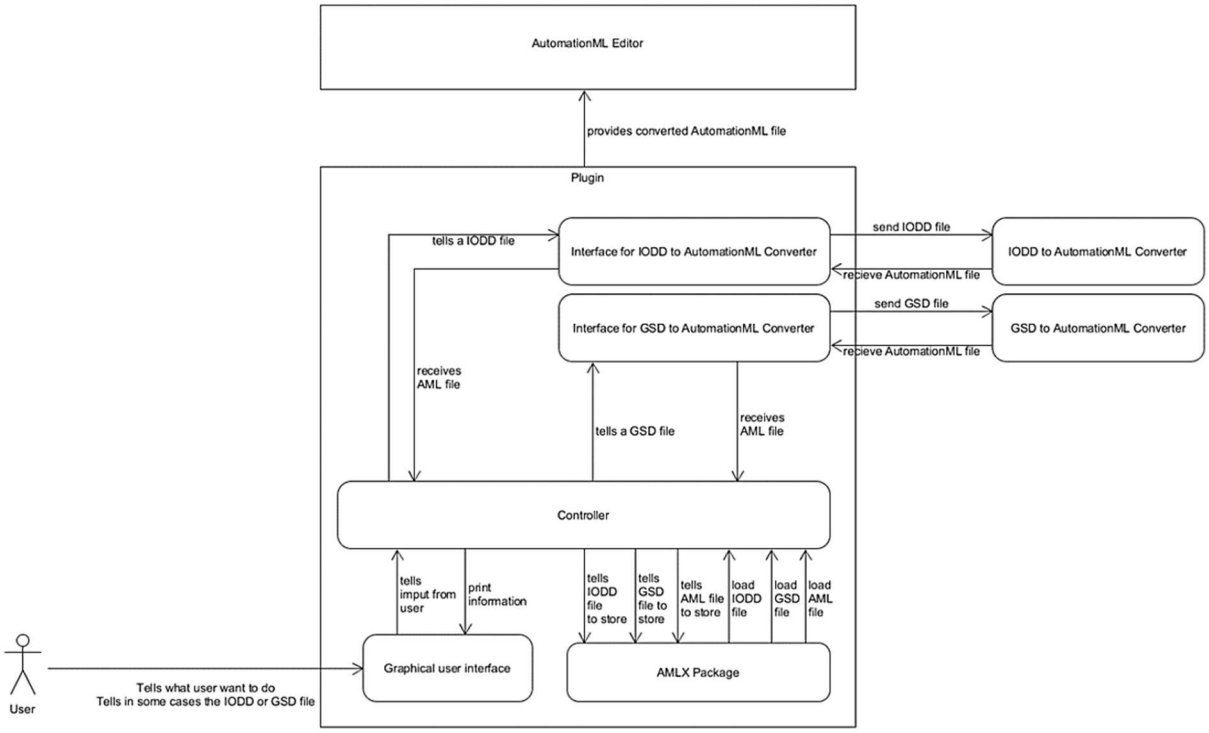

Figure 1 - System Design

The plugin was designed and developed in a Model-View-Control (MVC) architecture, which resembles a cycle. The user can use the plugin by accessing the GUI. However, the actions he performs in the GUI are not processed in the GUI but in the controller. The controller executes the changes, also called manipulations, in the background. The changes are then updated on the GUI, so that the user thinks that the changes were made directly in the GUI (compare figure 1).

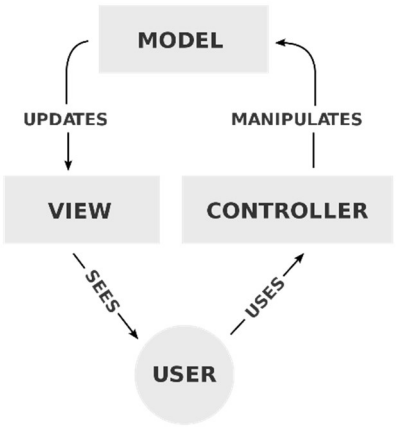

Figure 2 – MVC Architecture [1]

3.2. Usability concept

The criteria for good usability are:

- **Color scheme**:
  A good coloration can be created by an attractive choice of colors and their coordinated contrasts.

- **Design:**
  When designing the plugin, an appealing arrangement is important. It should be clearly recognizable which function is behind which visual elements and how the user can navigate through the plugin.

- **Intuitiveness:**
  Intuitiveness is a key component of good usability which means that the user understands the plugin without a long training period.

- **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.

Implementation guideline:
After carrying out the usability test, guidelines were developed, based on which the GUI improvements are to be made. Therefore, the GUI must meet the following guidelines:

- The same colors that are used in the AML editor should be used to create a consistent color scheme.

- Contrasts and frames should be used to emphasize input or fields of interaction and empty areas should be avoided to create an attractive and tidy design. The layout of the GUI should also match the style of the AML editor.

- The design and layout should be self-explanatory and reinforce the previous point in order to allow an intuitive use.

- The controls should be as close as possible to those of the AML editor, so that the user of the AML editor can find his way around faster.

Based on the criteria just defined and the developed guidelines the graphical interface is adapted and optimized. Functionality is very important but should not negatively influence the usability. Nevertheless, compromises must be made in terms of viability.

4. Subsystem specification

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

This module and its submodules have the task to improve the GUI to achieve a good usability. Furthermore, the module deals with bug fixing, because bugs are caused by errors in the GUI and therefore must be prevented there as well.

<MOD.001>	*Graphical User Interface*
System requirements covered:	*/NF10/, /NF20/, /NF30/, /NF40/, /NF50/, /NF60/, /NF70/, /NF80/, /NF90/, /NF100/, /F10/, /F20/, /F60/, /F70/, /F80/, /BUG10/, /BUG20/, /BUG30/*
Service:	• *Description in the submodules*
Interfaces:	• *Description in the submodules*
External Data:	• *Description in the submodules*
Storage location:	

4.1.1. <SUBMOD.001>: GUI customization

<SUBMOD.001>	*Graphical User Interface customization*
System requirements covered:	*/NF10/, /NF30/, /NF40/, /NF50/, /NF60/, /NF70/, /NF80/, /NF90/, /F60/*
Service:	• *Improve and customize the GUI design* • *Renaming of components to improve the intuitiveness* • *Improvement and adaptation of the design to the AML editor for recognition*
Interfaces:	• *No interfaces used*
External Data:	• *No external data required*
Storage location:	

4.1.2. <SUBMOD.002>: GUI simplification

<SUBMOD.002>	*Graphical User Interface simplification*
System requirements covered:	*/NF20/, /F20/*
Service:	• *Removing not necessary fields & buttons to achieve a better user experience and simpler structure of the program* • *Delete unused & confusing functionalities*
Interfaces:	• *Controller – GUI interface* • *Frontend and the corresponding backend*
External Data:	• *No external data required*
Storage location:	

4.1.3. <SUBMOD.003>: GUI extension & repair

<SUBMOD.003>	*Graphical User Interface extensions & repair*
System requirements covered:	*/F10/, /F30/, /F70/, /F80/, /BUG10/, /BUG20/, /BUG30/*
Service:	• *Functional enhancements for new requirements of functions on the GUI* • *Repair functions that do not work or do not work as intended* • *Error management for the uncaught bugs*
Interfaces:	• *Controller – GUI interaction* • *The possible actions from users that can cause errors* • *Areas with click effects*
External Data:	• *No external data required*
Storage location:	

4.2. <MOD.002>: Controller functionalities

This module is about the functional extensions for file formats, for the CAEX formats files of version 2.15 or 3.0 should be possible.

<MOD.002>	*Controller functionalities*
System requirements covered:	*/F40/, /F50/*
Service:	• *Output file format* • *Allow CAEX 2.15 or CAEX 3.0 as file format*
Interfaces:	• *Output path* • *Controller interface to save and create files*
External Data:	• *CAEX format rules*
Storage location:	

5. Technical Concepts

5.1. Persistence

Persistence is given by the package format. AMLX packages can be created and edited by the plugin. This format is also used in AutomationML and therefore it is possible to open the AMLX packages and use them in the editor.

5.2. User Interface

The graphical user interface (GUI) is the interface between user and program logic. The GUI allows the user to add new devices to the AutomationML Editor using the Modelling Wizard plugin.

5.3. Ergonomics

It is important for an ergonomic GUI to be intuitive. There are simple rules to follow to make the user experience as good and appealing as possible. This includes making sure that the design is appealing and that it is created in such a way that the user intuitively understands how to use it.

5.4. Communication with other IT-Systems

In the plugin there are use cases, for which external converter systems are integrated. These include the IODD converter and the GSD converter for AutomationML. With the converters IODD and GSD files can be converted into AML files to realize the functions of the plugin.

5.5. Deployment

It is not a stand-alone software that can be used without AutomationML. The plugin must be installed in the AutomationML editor using the plugin manager of AutomationML to install the .dll file.

5.6. Data Validation

The data check takes place in the background by the controller. This includes incorrect entries and missing information, which must be specified as mandatory information.

5.7. Exception Handling

The exception handling must be done to prevent errors caused by the user while using the GUI. Therefore “try-catch” blocks are used to prevent unwanted behavior of the program.

5.8. Internationalization

Since the language for the plugin and the user manual is English, the tool can be used internationally. But it is not possible to change the language individually, therefore English is obligatory.

5.9. Testability

To get an overview of the tests, the system test plan provides further information and the system test report contains their results. Overall, created AML file should be validated by the AML Component Checker.

5.10. Availability

The program is only distributed on GitHub and GitHub is the only possible source.

6. Figures

7. References

[1] „Wikipedia,“ 4 November 2020. [Online]. Available: https://en.wikipedia.org/wiki/Model%E2%80%93view%E2%80%93controller.

[2] I. -. DHBW, „SAS INF17C,“ 2019.