DHBW
Duale Hochschule
Baden-Württemberg
Stuttgart

MOD.002 Processes

(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)
	Graphical Designer	– Sophie Kirschner (inf21083@lehre.dhbw-stuttgart.de)

Change History

Version	Date	Author	Comment
0.1	28.04.2023	Fabian Kreuzer	Created
1.0	28.04.2023	Fabian Kreuzer	Finished

Scope

This module contains the processes in a more detailed way. It shows how the project is working in the backend and how the input from the user is validated and processed.

Glossary

The Modelling Wizard for Devices is a stand-alone software that can be used to create or modify devices and interfaces. It can also be used to import CAEX 2.15 and CAEX 3.0 files that will be converted to the AMLX (.aml; .amlx; .xml;) package.

Module Requirements

Requirements

- LF10: Import
- LF20: File validation
- LF30: Error handling
- LF70: Create device
- LF80: Export device

Module Context

This module includes the backend of the application. It is responsible for ensuring that the data in the graphical user interface is correct and also that all the required data is in place. The data entered is then processed by the controller and put into the correct structure. If any information for creating an AMLX file is missing, the user will be informed about it via a specific error message. Images are attached as an external file in the AMLX container. Interfaces, role class and all associated libraries are stored in the root-aml file. Afterwards the file can be exported. The controller is also responsible for allowing the import of AMLX files. If this is not possible the user will be informed.

Analysis

The main task of the processes is to open old files, create new ones and to load the standard libraries. When changing your options, the processes may load the given option into instances, so that the GUI knows what it has to display. It also holds the methods to search for GUID of a LoadedLibrary in instances. The save function might not be implemented due to time restrictions. While opening the files, it has to be considered that not all given files have a correct format or are corrupted, if one is the case it should be given the user as a feedback. We should also implement the option to not only load .aml and .amlx files but also .edz files. for this case, the converter class from the old project should be used. When creating new files we should give the user a SystemUnitClass as a base node.

Design

We desinged this as a library of static classes which each hold the corresponding functions.

Load holds everything what is needed to get all Roleclasses, InterfaceClasses and SystemUnitClasses out of any .aml or .amlx file. This class has only one public function which should return three different library objects. All the other private functions are only used to ensure everything is loaded. It has a byte array as an input.

Open is used when the user clicks on Open, it checks the file endings and then makes a byte array of the contents. These are send to load and after that are stored in instances as LoadedLibrarys.

New is used when the user creates a new file. It asks the user if he wants to discard unsaved changes if he has any. After that, we flush the LoadedLibrarys and create a new SystemUnitClass as a base.

Module Tests

Req.-ID	Test suite	Status
LF10	TS-002	success
LF20	TS-002, TS-003	success
LF30	TS-003	success

Req. ID	Test suite	Status
LF70	TS-001	success
LF80	TS-002	fail