DHBW
Duale Hochschule
Baden-Württemberg
Stuttgart

System Test Plan

(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)
Graphical Designer	– Sophie Kirschner (inf21083@lehre.dhbw-stuttgart.de)
Product Manager	– Maximilian Trumpp (inf21123@lehre.dhbw-stuttgart.de)

Change History

Version	Date	Author	Comment
0.1	11.03.2023	Michael Grote	Initial setup
0.2	16.03.2023	Michael Grote	Add features
0.3	04.04.2023	Michael Grote	Add testcases
0.4	16.04.2023	Michael Grote	Add testcases
0.5	08.05.2023	Michael Grote	Add file paths
1.0	09.05.2023	Michael Grote	Add Reviewer and Tester
1.1	10.05.2023	Michael Grote	Content addition

Table of Contents

1 Introduction

The STP (Software Test Plan) is used to validate the "Modeling Wizard" application. The Modeling Wizard is an application that can be used to create and edit a device. The created devices can be exported to an aml-file or an existing device can be imported from a file. The tests should check whether the customer's requirements for the software are fulfilled. The requirements are taken over here from the SRS (System Requirements Specification).

1.1 Scope

The tests check the fulfillment of the requirements from the SRS in the software. The tests check whether the functional and non-functional requirements are covered by the implemented functions. The document derived from the STP is the STR (System Test Report), which also lists the results.

1.2 Glossary

TC Testcase
TS Test suite
TD Test data
GUI Graphical User Interface
Req Requirement

1.3 Product Names and Attributes

The following test objects must be verified:

Ref.-Id.	Product Number	Product Name	Product Description
1	Build v1.0.0	Standalone Modelling Wizard	Windows Standalone Application with a GUI

2 Features

The following requirements must be verified, as long as they are not classified as “not to be tested”. This table shows the test coverage between functionality and test suite.

Req.-ID	Functionality	Priority	Test suite
LF11: Import	Imports file by absolute path	A	TS-002
LF12: File validation	Detect wrongly formatted imported files and throw an error to the user	B	TS-002, TS-003
LF13: Error handling	Handle errors (e.g. wrong user input)	A	T-S003

LF14 Edit device	Edit the attributes of a device	A	TS-001
LF15 Create device	Create an own device by the imports from the libraries	A	TS-001
LF16 Export device	Save created device into file	A	TS-002
NF11 GUI	Display a GUI and accept user input.	A	TS-001
NF12 Display device in a readable way	Display the attributes of selected device.	A	TS-001
NF13 Easy Mode	Only show a minimal view of attributes information	*not to be tested*	
NF14 Expert Mode	Show more information of an attribute to the user	*not to be tested*	
NF15 Portable	Program can be executed without installation	*not to be tested*	
NF16 Performance	Application should response instantly after user interaction	*not to be tested*	
NF17 Compatibility	The software should be executable on Windows 10 or higher	*not to be tested*	

3 Test Preparation Strategy

The creation of tests will be use case specific. Two main use cases can be identified, the new graphical user interface and file operations.

The graphical user interface is the first main use case. The GUI provides a view of the loaded device with input fields where the respective device data can be displayed and edited by the user. If interaction with the software by the user is not possible, the software would not fulfill its use case. In addition, the other use cases could not be properly tested and validated. For these reasons, this use case should be tested first.

Next, the file operations should be tested. Device files must be loaded, validated, and saved to ensure full functionality of the application for the user. Simply recreating them without the ability to export them would not meet all the requirements for the software. However, the user can still interact with the software without these functionalities.

Since with each input of user, in addition, external files an error input can take place, the error handling must be tested. In the case of errors, the user should be informed about the error and the cause of the error should be explained as well as possible. To test the error handling, it is necessary that the functionality exists in valid use. Therefore, the error handling tests should be tested last.

4 Test Execution Strategy

Most of the application's functions have been changed and should be tested. Since it is not possible to test all cases, we test the main functions. The test is divided into the following phases based on the preparation strategy:

1. Graphical User Interface
2. File operations
3. Error Handling

Since user interactions are necessary for the application to work, they must be tested first. This includes starting the program and executing the main functions of the application in the GUI. Since the software uses the same functions for the three menus "System Classes", "Interfaces" and "Role Classes", only the tests for the "System Classes" are performed. For a more detailed test of the software, it would be possible to extend the tests to the other two menus.

After the interaction with the user has been tested, the file operations can also be tested. The completion of the GUI testing is necessary because when testing the file operations, functionalities from the GUI are used. For testing the file operations, the test files from GitHub are needed (see Testfiles).

At last, the error handling can be tested. Here, erroneous input by the user, as well as erroneous file interactions are tested. Again, the test files stored in GitHub are referenced (see Testfiles).

5 Test Equipment

The following equipment must be available for testing:

- A computer with Windows 10 or higher
- The standalone Device Modelling Wizard software

6 Test Schedule and Budget

The testing of the application begins as soon as the application is completed. This makes it possible to make the necessary corrections quickly.

No budget is needed for the tests, as they are all performed by hand. Only the tester needs to get paid.

7 Test Planning

Testsuite	Test objective	Testplan Creator	Testplan Reviewer	Tester
TS-001	User interaction with Data	Michael Grote	Sophie Kirschner	Robin Ziegler
TS-002	File operations	Michael Grote	Sophie Kirschner	Robin Ziegler
TS-003	Error Handling	Michael Grote	Sophie Kirschner	Robin Ziegler

8 References

The Software Requirements Specification (SRS) can be found in GitHub (See https://github.com/robinziegler/TINF21C_Team4_Modelling_Wizard_Improvements/wiki/Software-Requirements-Specification-%5BSRS%5D).

The needed test files are also in GitHub (see https://github.com/robinziegler/TINF21C_Team4_Modelling_Wizard_Improvements/tree/master/PROJECT/STP/Testfiles).

9 Appendix: Testcases

9.1 Naming conventions

Test suite = <TS-**TS_number**>

Testcase = <TS-**TS_number**-**TC_number** >

Test data = <TD-**TS_number**-**TC_number** >

9.2 Tests

<TS-001> User interaction with Data

<TC-001-001> (Create new device)

Testcase-ID:	TC-001-001
Testcase-Name:	Create new device
Req.-ID:	NF11, NF12, LF14, LF15
Description	The test case verifies that the application can create a new device.

Test Steps		
Step	Action	Expected result
1	Open the Application Modelling Wizard.	Application starts without problems.
2	Create a new device by clicking “Create new File”.	In “System Classes” should be one device with the name “AutomationComponent”.
3	Click on the “Add System Unit Class”-Button in “System Classes”. Expand the tree view and check one device and then click “Add”.	A dialog should be opened. After clicking on “Add” the selected devices should be added into “System Unit Class” and the dialog should be closed.

<TC-001-002> (Edit attribute of device)

Testcase-ID:	TC-001-002
Testcase-Name:	Edit attribute of devices
Req.-ID:	NF11, NF12, LF14
Description	The test case verifies that the application can create a new device.

Test Steps		
Step	Action	Expected result
1	Open the Application Modelling Wizard.	Application starts without problems.

2	Create a new device by clicking “Create new File”.	In “System Classes” should be one device.
3	Open the “AutomationComponent”. Open the expander with the name “IdentificationData” Now change the value of Manufacturer in the right grid. Then select another attribute in the grid and change the value of this attribute.	The user input should be accepted and displayed correctly in the grid.

<TC-001-003> (Add attachment)

Testcase-ID:	TC-001-003
Testcase-Name:	Add attachment to device
Req.-ID:	NF11, NF12, LF14
Description	The test case verifies that the application can add a new attachment to a device.

Test Steps		
Step	Action	Expected result
1	Open the Application Modelling Wizard.	Application starts without problems.
2	Create a new device by clicking “Create new File”.	In “System Classes” should be one device with the name “AutomationComponent”.
3	Open the “Attachments” in the menu bar.	The Attachments menu item should now be highlighted with a blue line.
4	Now press "Add Attachment" and select the file.	The file explorer will be opened. After selecting the file, the file name should be listed in the application.

Test data: TD-001-003		
Data set	File	Validation
1	139059_Festo_Automatisierung_Prozessventile_WhitePaper_DE142460_202005_V01.pdf	valid

<TC-001-004> (Change the theme)

Testcase-ID:	TC-001-004

Testcase-Name:	Change the theme
Req.-ID:	NF11, NF12
Description	The test case verifies that the application can change the colour theme.

Test Steps		
Step	Action	Expected result
1	Open the Application Modelling Wizard.	Application starts without problems.
2	Create a new device by clicking “Create new File”.	In “System Classes” should be one device with the name “AutomationComponent”.
3	Click on the “Options” in the menu bar and click on “Darkmode” or “Lightmode” depends on your actual theme.	The theme of the application should be changed.
4	Open the “System Classes”, “Interfaces”, “Role Classes” and “Attachments” and compare the design with the expected design pattern.	The colour theme should be used in every menu. In Darkmode the text colour should be white and the background light grey. In lightmode the background should be white and the text colour should be black.

<TC-001-005> (Load a new valid library)

Testcase-ID:	TC-001-005
Testcase-Name:	Load a new valid library
Req.-ID:	NF11, LF15
Description	The test case verifies that the application can load a new aml-library.

Test Steps		
Step	Action	Expected result
1	Open the Application Modelling Wizard.	Application starts without problems.
2	Create a new device by clicking “Create new File”.	In “System Classes” should be one device with the name “AutomationComponent”.
3	Click “Libraries” in the menu bar, then “Add library”.	The explorer should be opened.
4	Select the library file to import it.	The library should be loaded into the Application and displayed in the list under “Libraries” in the menu bar.

Test data: TD-001-004		
Dataset	File	Validation
1	valid/IndustrialSensorLibrary_v1_0_0.aml	valid

<TS-002> File operations

<TC-002-001> (Loading of a valid file with validation)

Testcase-ID:	TC-002-001
Testcase-Name:	Loading of a valid file with validation
Req.-ID:	LF11, LF12, NF12
Description	The test case verifies that the application can load a valid file.

Test Steps		
Step	Action	Expected result
1	Open the Application Modelling Wizard.	Application starts without problems.
2	Select a valid input file for the validation, by selecting “Open File”, and then choose the file in explorer.	The validation is executed successfully, and the file is loaded completely and correctly without error message.
3	Check that the data has been interpreted correctly Compare the expected data with the data in “System Unit Class” and their “Attributes”, also compare the data in “Interfaces” and “Role Classes”.	All data should be displayed readable and correctly.

Test data: TD-002-001		
Dataset	File	Validation
1	valid/MURR.4000-73000-0200000.amlx	valid
2	valid/MVK MPNIO DI6 DO6 IOL2 IRT PP-55516.amlx	valid

<TC-002-002> (Export of a valid device in a file with validation)

Testcase-ID:	TC-002-002
Testcase-Name:	Export of a valid device in a file with validation
Req.-ID:	LF11, LF12, NF12, LF14, LF16
Description	The test case verifies that the application can export a valid file.

Test Steps		
Step	Action	Expected result
1	Open the Application Modelling Wizard.	Application starts without problems.
2	Select a valid input file for the validation, by selecting “Open File”.	The validation is executed successfully, and the conversion is completed correctly without error message.

3	Edit some attributes in “System Unit Class”, be sure to add some new attributes and change some old ones.	Changes are displayed correctly. In the title bar the text changed from “saved” to “unsaved”.
4	Click on “File” and select “Save”, select location in the file explorer and save file.	Valid file can be saved without errors and filename is generated automatically by manufacturer and product code.
5	Open the new file in the Application and check if the changes were applied correctly and the file is still valid.	File opened without error and changes of the attributes are displayed correctly.

Test data: TD-002-002		
Dataset	File	Validation
1	valid/MURR.4000-73000-0200000.amlx	valid
2	valid/Murrelektronik_7000-40021-6340500.amlx	valid
3	valid/MVK MPNIO DI6 DO6 IOL2 IRT PP-55516.amlx	valid

<TS-003> Error Handling

<TC-003-001> (Loading of an invalid file with validation)

Testcase-ID:	TC-003-001
Testcase-Name:	Loading of an invalid file with validation
Req.-ID:	LF11, LF12, LF13
Description	The test case verifies that the application can validate a file and find invalid files by opening.

Test Steps		
Step	Action	Expected result
1	Open the Application Modelling Wizard.	Application starts without problems.
2	Select an invalid input file for the validation, by selecting “Open File”, and then choose the file in explorer.	The validation is executed successfully, and the file is not loaded. The User get a message about the error.

Test data: TD-003-001		
Dataset	File	Validation
1	invalid/7000-40041-5770150.amlx	invalid
2	invalid/7030-12361-1261000.amlx	invalid
3	invalid/MURR.8000-88010-3570500.amlx	invalid

<TC-003-002> (Export of an invalid device in a file with validation)

Testcase-ID:	TC-003-002
Testcase-Name:	Export of an invalid device in a file with validation
Req.-ID:	LF11, LF12, LF13, LF14, LF16
Description	The test case verifies that the application can handle a wrong input from the user and that it informs the user right.

Test Steps		
Step	Action	Expected result
1	Open the Application Modelling Wizard.	Application starts without problems.
2	Select a valid input file for the validation, by selecting “Open File” or create a new file by “New file”	The validation is executed successfully, and the conversion is completed correctly without error message.
3	Delete the manufacturer of the first element in “System Unit Class” under “IndentificationData”.	An orange warning sign appears at the upper right edge of the application. Hovering over the ICON displays the error message.
4	Click on “File” and select “Save”.	The file cannot be saved, and the application should show a message with the reason.

Test data: TD-003-002		
Dataset	File	Validation
1	valid/Murrelektronik_7000-40021-6340500.amlx	valid