APPLICATION FOR FORMAL TESTING OF PLC PROGRAM

O. Sýkora[1], J. Arm[2]

[1]Brno University of Technology, Department of control and instrumentation, Czech Republic

[2]Brno University of Technology, Department of control and instrumentation, Czech Republic

E-mail: 221017@vut.cz, arm@vut.cz

Abstract - The paper deals with developing application for testing of selected errors contained in Siemens TIA Portal V16 PLC Project. The application is being developed for the company ICE Industrial Services. The application should find bugs that programmers or PLC code generators make, but TIA Portal cannot detect them defaultly. In this moment, application is able to open or find opened TIA Portal project and load its important parts to my structure – *ProjectBlocks* structure. Application can also find some errors described in Chapter 2.

Keywords - C#, TIA Portal, TIA Portal Openness, Windows form application

1. INTRODUCTION

Code validation testing (whether desktop application development code or PLC code) is an important activity, the underestimation of which can lead to increased costs development. If the software is not tested, there is a good chance that when it continues to operate in near future mistake. At best, the problem still occurs in development phase, in the worst case for a customer who notices defects or errors and will request a repair or refund. [1]

There are 2 testing technique areas: Static software testing and dynamic software testing. If static software testing techniques is used, tester do a visual control of PLC program – visually check, if code doesn´t contain any visible error. Also, can find a mistake or irregularity in the project documentation and discuss it with his colleagues. In other words, static software testing is used for error prevention. On the contrary, dynamic software testing techniques usually running the code to determine what kind and how many errors are contained in program. There are two approaches: Black box testing techniques and white box testing techniques. [2]

Black box testing means that tester may not understand at all how the tested program code works. Probably he is not allowed to even see a code. His job is to know the purpose of the program and to be able to distinguish between its correct and incorrect working. He is typing data to the aplication or (in PLC programming case) i.e., do an IO check – how controlled machine is working when something he pulls, activates the sensor, etc. [2]

When white box testing approach is selected, tester is code-knowledgeable person – in most cases programmer himself and he is trying diverse edge cases to error-testing a program such as statement coverage, branch coverage, condition coverage or basic path testing. [2]

There are lot of approaches and testing techniques being used in software development industry, but only four testing levels in project develing process. Unit testing is a level of SW testing, where individual units (such as station funcions or function blocks) being tested. The purpose is to validate that each unit of the software performs as designed. Integration testing is a level of the SW testing process where individual units are combined and tested as a group. The purpose of this level of testing is to expose faults in the interaction between integrated units. System testing is a level of the software testing process where a complete, integrated system is tested. The purpose of this test is to evaluate the system’s compliance with the specified requirements. And finally, acceptance testing is a level of the software testing process where a system is tested for acceptability. The purpose of this test is to evaluate the system’s compliance with the business requirements and assess whether it is acceptable for delivery. [3]

In the field of PLC program testing, there are several approaches in use: Manual testing – basically dynamic white box testing, where programmer can watch variables behavior in the program run, automatic testing – testing cases are generated i.e., by SCADA system. Also, formal testing techiques can be used. [4]

2. TIA PORTAL COMMON ERRORS

As the abstract says, my paper deals with developing application for testing of selected errors in PLC (ladder) code. This PLC code is written by PLC programmers in one smaller company in my city and in most cases containing lot of errors or bugs of various relevancy such as:

- Reading Temp variable before writing to it
- Not named networks
- Wrong used sensor symbolics (i.e., sensor from Station 120 used in Station 180)
- Same variable assignments on multiple coils
- For more instances of used function blocks, timers, triggers, etc. used same DB variable
- Not used variable, which is defined in the interface of the PLC block
- Too many instructions in one network
- Used bad merkers
- Default name of HW components
- Not assigned partner in PN topology

Version of TIA Portal: V16

3. DIAGRAM

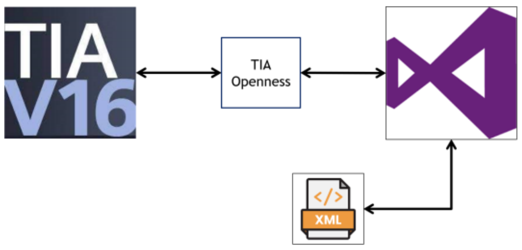


Figure 1: Simple diagram of application, communicating with TIA Portal and exporting blocks via TIA Portal Openness

4. HOW THE APPLICATION WORKS

The application for PLC code testing is being developed in Visual Studio 2019, C# programming language and as a simple windows form application. Application communicates with opened TIA Portal project via TIA Portal Openness. TIA Portal Openness is the API for WinCC and STEP7 in the TIA Portal, which enables users to create their own TIA Portal controlling application, or enabling automate their engineering tasks, or read info about HW configuration, PLC blocks, HMI devices, etc. Application can open TIA Portal or TIA Portal project or recognize and connect to opened TIA Portal project and load it into a TiaProject class instance in my application, which is provided by Openness and inform the user about connected project and show its name.

The main problem of Openness is that it cannot access individual networks and network parts, so the Openness is for most of the problems described in Introduction unusable. So, there must be another way to access networks and read specific information about them and this is can be realize via XML prescription of PLC blocks.

After the successufull connection is made, application will clear special folder, and then recursively export all PLC blocks from TIA Portal project to this folder, again by Openness. When this is done, application starts read info and taking it to my special structure of PLC blocks, already containing the info about all networks, parts, variable accesses, wires (ladder diagram) and FB/FC calls.

After all blocks are loaded into *ProjectBlocks* structure, application starts error finding algorithms and taking them into lists of errors. Every error, found in the structure, is recorded, and shown in the list view. Application enables user to download list view to text file. Error is identified by its name, description, project, PLC block and network in which it occurs.

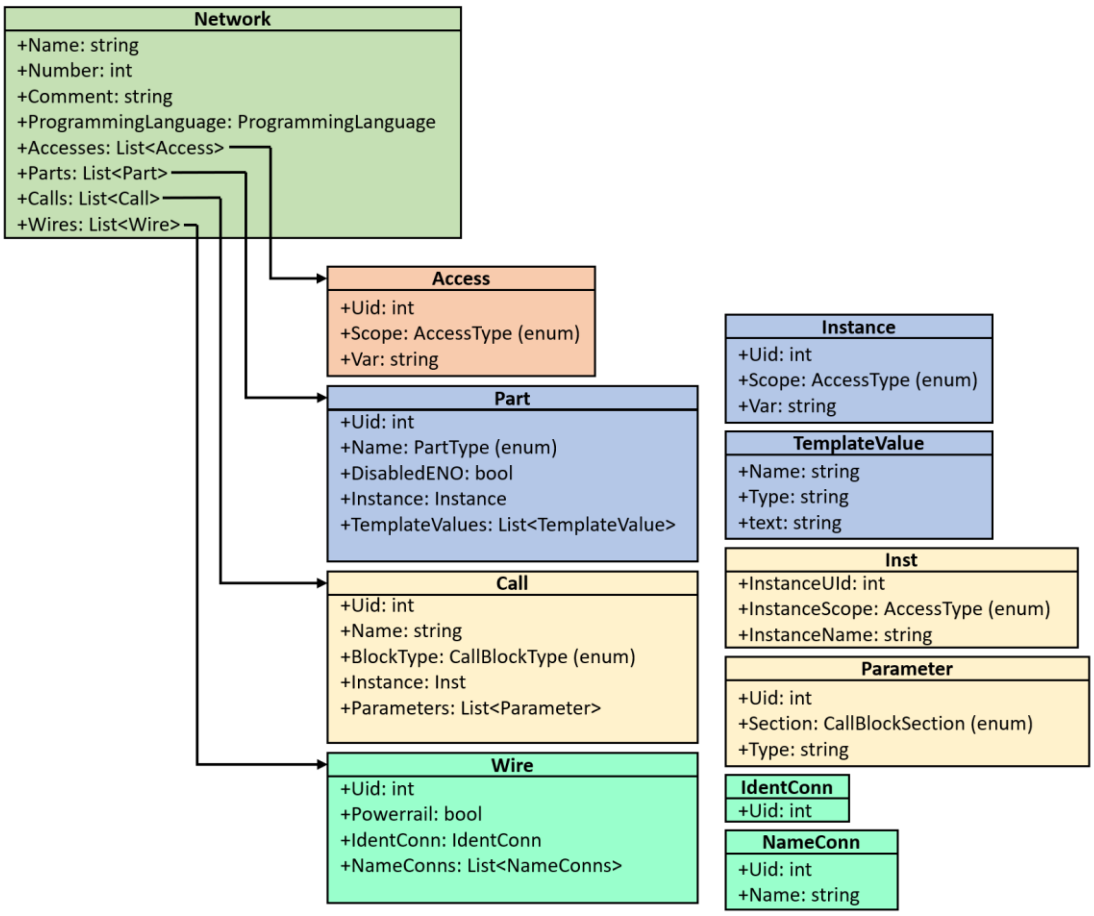
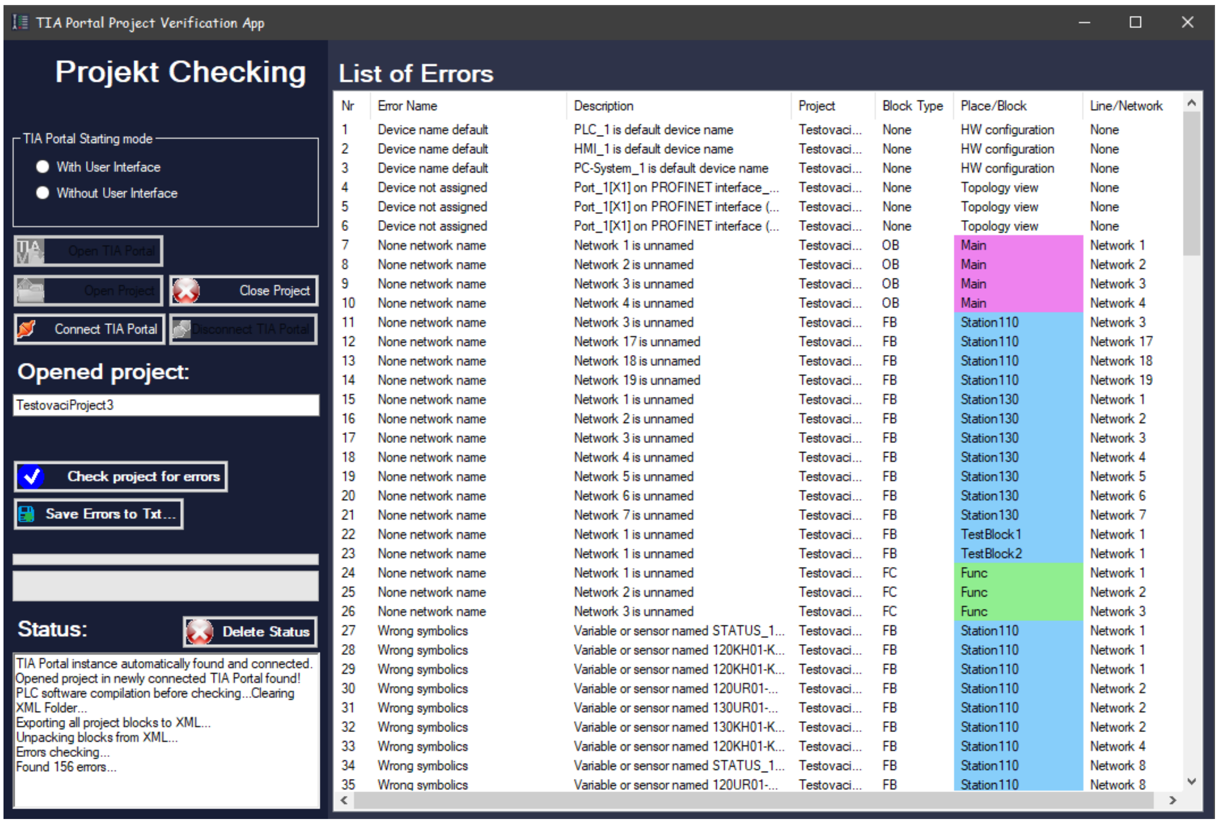


Figure 2: Network class of PLC block in *ProjectBlocks* structure



TestovaciProject3

TIA Portal instance automatically found and connected.
Opened project in newly connected TIA Portal found!
PLC software compilation before checking...Clearing XML Folder...
Exporting all project blocks to XML...
Unpacking blocks from XML...
Errors checking...
Found 156 errors...

Nr	Error Name	Description	Project	Block Type	Place/Block	Line/Network
1	Device name default	PLC_1 is default device name	Testovaci...	None	HW configuration	None
2	Device name default	HMI_1 is default device name	Testovaci...	None	HW configuration	None
3	Device name default	PC-System_1 is default device name	Testovaci...	None	HW configuration	None
4	Device not assigned	Port_1[X1] on PROFINET interface_...	Testovaci...	None	Topology view	None
5	Device not assigned	Port_1[X1] on PROFINET interface (...	Testovaci...	None	Topology view	None
6	Device not assigned	Port_1[X1] on PROFINET interface (...	Testovaci...	None	Topology view	None
7	None network name	Network 1 is unnamed	Testovaci...	OB	Main	Network 1
8	None network name	Network 2 is unnamed	Testovaci...	OB	Main	Network 2
9	None network name	Network 3 is unnamed	Testovaci...	OB	Main	Network 3
10	None network name	Network 4 is unnamed	Testovaci...	OB	Main	Network 4
11	None network name	Network 3 is unnamed	Testovaci...	FB	Station110	Network 3
12	None network name	Network 17 is unnamed	Testovaci...	FB	Station110	Network 17
13	None network name	Network 18 is unnamed	Testovaci...	FB	Station110	Network 18
14	None network name	Network 19 is unnamed	Testovaci...	FB	Station110	Network 19
15	None network name	Network 1 is unnamed	Testovaci...	FB	Station130	Network 1
16	None network name	Network 2 is unnamed	Testovaci...	FB	Station130	Network 2
17	None network name	Network 3 is unnamed	Testovaci...	FB	Station130	Network 3
18	None network name	Network 4 is unnamed	Testovaci...	FB	Station130	Network 4
19	None network name	Network 5 is unnamed	Testovaci...	FB	Station130	Network 5
20	None network name	Network 6 is unnamed	Testovaci...	FB	Station130	Network 6
21	None network name	Network 7 is unnamed	Testovaci...	FB	Station130	Network 7
22	None network name	Network 1 is unnamed	Testovaci...	FB	TestBlock1	Network 1
23	None network name	Network 1 is unnamed	Testovaci...	FB	TestBlock2	Network 1
24	None network name	Network 1 is unnamed	Testovaci...	FC	Func	Network 1
25	None network name	Network 2 is unnamed	Testovaci...	FC	Func	Network 2
26	None network name	Network 3 is unnamed	Testovaci...	FC	Func	Network 3
27	Wrong symbolics	Variable or sensor named STATUS_1...	Testovaci...	FB	Station110	Network 1
28	Wrong symbolics	Variable or sensor named 120KH01-K...	Testovaci...	FB	Station110	Network 1
29	Wrong symbolics	Variable or sensor named 120KH01-K...	Testovaci...	FB	Station110	Network 1
30	Wrong symbolics	Variable or sensor named 120UR01-...	Testovaci...	FB	Station110	Network 2
31	Wrong symbolics	Variable or sensor named 130UR01-...	Testovaci...	FB	Station110	Network 2
32	Wrong symbolics	Variable or sensor named 130KH01-K...	Testovaci...	FB	Station110	Network 2
33	Wrong symbolics	Variable or sensor named 120KH01-K...	Testovaci...	FB	Station110	Network 4
34	Wrong symbolics	Variable or sensor named STATUS_1...	Testovaci...	FB	Station110	Network 8
35	Wrong symbolics	Variable or sensor named 120UR01-...	Testovaci...	FB	Station110	Network 8

Figure 3: Application interface

If you have opened TIA Portal project, you can simply run the application. Application will automatically connect to your opened project. You can open another project in application too. The found project name is displayed. The all you need to do is click on the *Check project for errors* button and full progress of error finding will be launched. On its end, results will be displayed in the list of errors to the right on the figure 2. You can save this list to txt file.

5. CONCLUSION AND WORK PROGRESS

My work's will be purely oriented in the software field. Practically, I work only with TIA Portal, Openness, Visual Studio, XML and text editors. My work is intended for company usage.

Currently, application is almost done. All error searching algorithms are implemented and my work is focusing on improving application's reliability and stability – trying to find exception throw cases, having meeting with company consultant and make testing cases in TIA Portal.

The benefits of this work, which I consider as important, are *ProjectBlocks* structure, object-oriented application and, of course, gained experiences with TIA Portal Openness. The biggest benefit is application functionality itself – it allows programmers to detect their mistakes, which TIA Portal can't detect defaulty by compiling. This role is very important in terms of PLC code testing and finding out where the error is occurred. In company, for which I am creating this application, are also being strictly used their own datablock variables and PLC tags which allow me to implement Wrong symbolics error searching algorithm.

The main disadvatage of my application is relative long error finding time, expecially if there is a huge project with lots of blocks, where each block contains over 100 networks. In this case, error finding time can reach 2-3 minutes, including blocks export to XML, blocks XML loading into *ProjectBlocks* structure and error searching itself.

Work progress

I started working on this application at the end of January 2022 and in first three weeks of work I was learning TIA Portal Openness and finding its functionalities, which can be important and crucial for my work. I found out, that Openness is unable to approach networks and interfaces of PLC blocks, but is able to export PLC blocks to XML files. And in this block XML file, networks and interface are already contained. Then I made a structure (which Networks part is shown up in fig. 2) and method, which contains algorithms of loading important parts of XML file to this structure. After that I was able to do error searching algorithms. In case of HW configuration errors, I didn't even need to use *ProjectBlocks* structure, in case of SW errors, *ProjectBlocks* structure was necessary.

ACKNOWLEDGMENT

This work was supported by Brno University of Technology and was carried out with the support of the project FEKT-S-20-6205 Research in automation, cybernetics and artificial intelligence for Industry 4.0.

REFERENCES

[1] Sharma, L. (2021, July 7). *What is software testing and ways of software testing.* TOOLSGA. Retrieved March 11, 2022, from https://toolsqa.com/software-testing/software-testing/

[2] Nidhra, S. (2012). Black Box and White Box Testing Techniques – A Literature Review. *International Journal of Embedded Systems and Applications*, *2*(2), 29–50. https://doi.org/10.5121/ijesa.2012.2204.

[3] *Software Testing Levels*. (2020, September 16). SOFTWARE TESTING Fundamentals. Retrieved March 11, 2022, from https://softwaretestingfundamentals.com/software-testing-levels/

[4] Fernández, B., Blanco, E., & MEREZHIN, A. (2013, October). *Testing & verification of PLC code for process control*. ICALEPCS, San Francisco, USA, from https://www.researchgate.net/publication/270279486_Testing_verification_of_PLC_code_for_process_control