Article

Not peer-reviewed version

Strategies for Software and Hardware Compatibility Testing in Industrial Controllers

Marcus Rothhaupt , Lucas Vogt * , Leon Urbas

Posted Date: 31 January 2024

doi: 10.20944/preprints202401.2239.v1

Keywords: compatibility; test automation; industrial controller

Article

Strategies for Software and Hardware Compatibility Testing in Industrial Controllers

Marcus Rothhaupt, Lucas Vogt * and Leon Urbas

TU Dresden, 01217 Dresden, Germany; marcus.rothhaupt@mailbox.tu-dresden.de (M.R.); leon.urbas@tu-dresden.de (L.U.)

* Correspondence: lucas.vogt@tu-dresden.de; Tel.: +49-351-463-36410

Abstract: Mass customization, small batch sizes, high variability of product types and a changing product portfolio during the life cycle of an industrial plant are current trends in the industry. Due to an increasing decoupling of the development of software and hardware components in an industrial context, compatibility problems within industrial control systems arise more and more frequently. In this publication, a strategy concept for compatibility testing is derived and discussed by means of literature review and applied research. This 4-phased strategy concept identifies incompatibilities between software and hardware components in the industrial control environment and enables test engineers to detect problems at an early stage. By automating the compatibility test on an external I-PC, the test can be run both when new software is installed on the industrial controller and when the controller is restarted. Thus, changes to the components are constantly detected and incompatibilities are avoided. Furthermore, early incompatibility detection can ensure that a system remains permanently operational. Based on a discussion, additionally strategies are identified to consolidate the robustness and applicability of the presented concept.

Keywords: compatibility; test automation; industrial controller

1. Introduction

Modern trends in manufacturing are characterized by mass customization, small batch sizes, high variability in product types, and a changing product portfolio during the life cycle of an industrial plant [1]. These trends imply more complex plants [2] that support changes in physical layout, including major engineering upgrades. The complexity of plants, including automation hardware and automation software, is increasing. As the percentage of system functionality realized by software increases, concepts to support automation engineers in dealing with this complexity are urgently needed [3].

Automated testing can help minimize the resources required for software development. However, changes to the software necessitate re-evaluation of functionality through testing. To reduce resource consumption, existing relevant tests can be re-run after ensuring their compatibility with the software after the changes [4]. If a software or its environment are changed, it is necessary to check, on the one hand, whether the desired function is fulfilled and, on the other hand, whether there are any unwanted changes or side effects [5].

In the proposed concept, the target and actual state of the software and hardware components is an essential part of the compatibility test. A stringent test procedure, which can always be repeated in exactly the same way, forms the framework for the concept and is also presented in this paper.

As a result of the compatibility test, the concept offers an overview of found incompatibilities and shows possible reactions.

The concept was tested and evaluated on a module of the P2O-Lab [6] of the TU Dresden . The results met the concept requirements, allowed the detection of incompatibilities and where therefore published as a German preprint [7].

2. Methods and State of the Art

The analysis of the requirements for the proposed concept is backed by a theoretical examination of the existing compatibility testing strategies in virtual commissioning (VC) and cyber security (CS). These strategies serve as a valuable foundation for developing a compatibility testing concept and are described in the following sections of this article (see sections 2.1 - 2.4).

The primary criterion for inclusion in compatibility testing, for both VC and CS, is the presence of a model to test against. Consequently, only two test strategies meet this criterion and are classified as essential for compatibility testing.

In the following section the most important literature findings, which were found to be most beneficial to the proposed concept, are highlighted. From virtual commissioning the testing strategies Software-in-the-Loop (SiL) and Hardware-in-the-loop (HiL) were found to be a good source of guidance for the development of the proposed concept. The findings from current literature developments are highlighted in chapter 2.1 and chapter 2.2). The other major literature topic which was used to influence the concept development was cyber security. The test strategy of anomaly-based detection (see chapter 2.4) from the topic of cyber security, was found to be connected to an approach which could be used for the proposed concept.

2.1. Software-in-the-loop

In the SiL approach, a virtual PLC is instantiated to test the automation code associated with the behavior models in the simulation layer [8].

This approach makes it possible to integrate software components with an environment simulation [9]. In addition, this approach enables very fast testing of different scenarios and control algorithms and their flexibility.

The costs for implementing an SiL environment are around sixty times lower than for a hardware-in-the-loop (HiL) environment. The SiL environment can be available to any developer, while separate equipment is required for HiL [9].

SiL tests are carried out by running the software on normal PC hardware, which makes it possible to identify the most important errors in the functional area. However, the compiler and the processor of a PC may behave differently than on the final automation platform[10].

2.2. Hardware-in-the-loop

In the HiL approach, a real physical PLC is connected to a simulation layer that executes the system's behavior models.

All VIBN processes are based on a virtual model that is connected to a PLC. In the case of a hardware-in-the-loop (HiL) simulation, the PLC is a real hardware controller [11]. Consequently, it is possible to carry out the VIBN with the PLC, which is then integrated into the production system. According to Mazza [10], this is particularly interesting for:

1. Validation of PLC control strategies based on a virtualized environment with the ability to represent the expected dynamics of the real machine
2. Improvement or comparison of real-world measured data with simulated data (e.g. from virtual sensors)
3. Support operators during real machine operation through simulated predictions or diagnostics fed by a 'digital twin' with real data from the field.

2.3. Use of SiL and HiL for the concept

To summarize, the following reasons can be found why the VIBN strategies SiL and HiL are very useful:

1. Control strategies can be virtually validated without endangering human lives or machines.

2. Costs can be reduced thanks to the possibility of debugging (error correction could occur too late during the design process)
3. Operators can familiarize themselves with the control systems, including those under construction, thanks to the creation of virtual systems.
4. Errors can be found within a few minutes with the help of ’virtual time’ through simulation.

2.4. Anomaly-based detection

Anomaly-based detection uses statistical methods and artificial intelligence to detect unknown attacks [12]. Ourston et al. [13] presented an approach that uses hidden Markov models to detect complex cyber attacks. This method is able to address the problem of multi-stage attacks. Experimental results have shown that this method is more effective than classical machine learning techniques, such as decision trees and artificial neural networks.

Mukkamala et al. [14] developed a method for detecting attacks using K next neighbors algorithms (KNN) and support vector machines (SVM). KNNs and SVMs were used to create the classifiers based on a list of features. Experimental results have shown that KNNs and SVMs are able to detect anomalies and known intruders. Pan et al. [15] developed a hybrid method for detecting attacks by combining KNNs and decision tree algorithms. Experimental results have shown that KNNs can detect DoS and probing attacks more effectively than detecting unauthorized access from a remote machine and authorized access to local superuser attacks.

Zhang et al. [16] developed a method based on random forests to detect network intrusions. This method was demonstrated on an intrusion detection data-set. The experimental results have shown that the proposed method can achieve a high detection rate with a low false positive rate.

Gaddam et al. [17] developed an anomaly detection approach using cascading K-Means clustering and ID3 decision tree learning algorithms. This method was used to analyze a data-set of network anomalies. Experimental results have shown that the detection accuracy is up to 96.24 % with a false positive rate of 3 %.

Liao and Vemuri [18] developed a classifier for intruder detection using the k-nearest neighbor (kNN) algorithm. This method was used to classify the behavior of programs as normal or intrusive. Experimental results have shown that the kNN classifier can effectively detect attacks with a low false positive rate.

Sabhnani and Serpen [19] analyzed an intrusion detection data-set using a set of machine learning algorithms. The data-set includes four types of major attacks, including probing, DoS, user-to-root, and remote-to-local attacks. Simulation results have shown that certain classification algorithms are more effective for a particular attack category.

Lee et al. [20] introduced an attack detection method based on cluster analysis to proactively detect DoS attacks. A hierarchical clustering algorithm was used to analyze a data-set for attack detection. Experimental results have shown that this method is capable of detecting DoS attacks.

3. Proposed Concept

For the proposed concept, SiL approach, based on [8] and HiL approach, based on [11] strategies from VC highlight the crucial role of a pre-established model in conducting effective tests. Additionally, from the field of CS, the strategy of anomaly-based detection, as described by Ourston et al. [13], emphasizes the use of a predefined model to detect deviations and potential attacks.

Based on this insight, it becomes evident that a model, referred to as the target state, is fundamental in compatibility testing. This target state encompasses the intended software and hardware configurations for compatibility testing.

In contrast to the target state, akin to the SiL or HiL strategies, there is the system to be tested, whether simulated or the physical programmable logic controller (PLC), to which the test is applied. In the proposed context of compatibility testing, this system under test is referred to as the actual state. It represents the current state of the hardware components connected to the PLC and the state of the software running on these devices. These two main points, the determination of a target and actual state, form the foundation of the compatibility testing concept.

The model checking and anomaly-based detection strategies from VC and CS can be adapted for use in the context of software upload and restart by introducing an additional component external to the PLC. This additional component takes on the role of monitoring the PLC and automatically initiating a compatibility check whenever a software update is pending or the PLC undergoes a restart. In the proposed concept, as outlined in chapter 3, this external component is an industrial pc (I-PC). The I-PC runs a test script responsible for managing the software upload to the PLC and monitoring the software restart process of the PLC.

The proposed concept consists of four phases (see Figure 1). Phase one conducts an automated self-test on the PLC connected to a test I-PC. This ensures basic PLC operation requirements, like CPU and I/O modules presence, memory checks, and power availability (see Chapter 3.1).

Phases two and three determine the actual and target state of software and hardware components in the system. First the actual state (see Chapter 3.3) of the system is determined, then the determination of the target state (see Chapter 3.2) follows. It is important to note, that the proposed concept is not generally applicable. The proposed concept is based on the TIA Openness API as foundation for the determination of the actual and target state.

In the fourth phase, the test compares target and actual states to identify differences and categorizes incompatibilities in error detection tables (see Chapter 3.5).

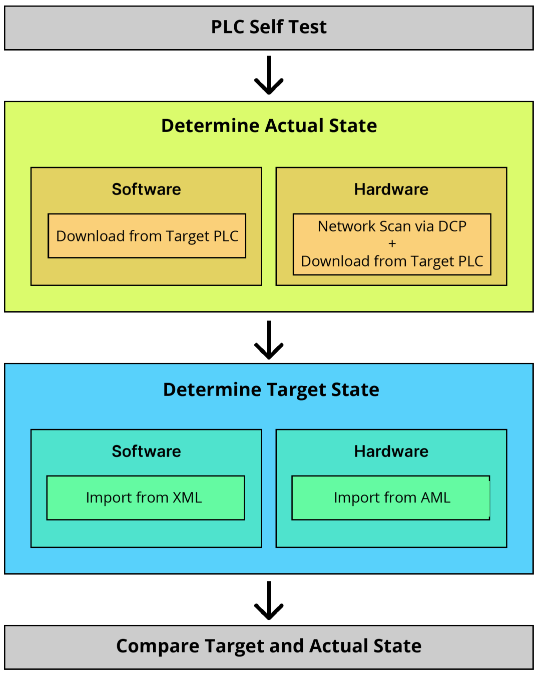

Figure 1. Proposed compatibility test sequence.

3.1. PLC Self Test

PLC self-test includes self-testing and diagnosis of a PLC under test. As required by DIN EN 61131-2, manufacturers of PLC systems must provide means for self-testing and diagnosing the operation of these systems. Furthermore, the self-test must allow a statement about the proper condition of a PLC system.

The PLC self-test according to DIN EN 61131-2 must provide diagnostic means to perform the following actions:

1. monitoring of the application program (watch dogs);
2. checking the integrity (freedom from errors) of the memory;

3. checking the correctness of data exchanged between memory, processing unit and I/O modules;
4. checking the power supply of the system;
5. monitoring the state of the main processing unit.

The output of the PLC self-test is essential to determine the suitability of the PLC for compatibility testing. The self-test provides the basis for meeting the hardware requirements of the compatibility test. A system which is not in proper condition, i.e. which does not pass the self-test, cannot be used as part of the concept for testing the compatibility between software and hardware.

3.2. Export & Import of Target State

Before the target state can be imported to check compatibility with the new hardware, the files required for the import must first be obtained. Depending on the use case, the requirements for importing the target state differ.

Use Case A - Determination upon PLC Restart

For use case A, the target state is imported from stored data on the test PC, representing the last known actual state before PLC restart. To ensure an automated sequence, the test PC retains the last actual state, making it available for compatibility tests after a PLC restart. A continuous ping between the test script and the PLC detects restarts, triggering automatic compatibility checks. See Figure 2 for the relevant components. Since the target state is already on the I-PC during a restart, it's simply loaded by the test script.

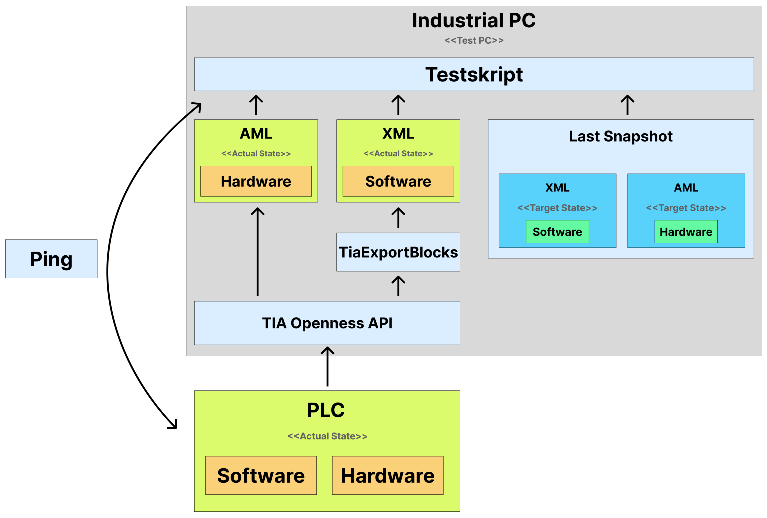

Figure 2. Determination of the target and actual state at restart.

Use Case B - Determination upon PLC Update

To install new software on the PLC, the target state comes from user-provided update data. This data typically includes a TIA project file, which is first opened using the TIA Openness API. The open-source software TiaExportBlocks [21] extracts variable tables in XML format from the TIA project and exports them to an XML file. The hardware topology is exported as an AML file [22] using the CAx export function of the TIA Openness API.

This process results in software data in XML format and hardware data in AML format (CAEX standard) [23].

The test script now has the target state for the software and hardware, which is essential for the compatibility check. The CAEX format in AML uses the PLCopen XML standard [24] for machine-readable hardware topology.

By determining the target state, the necessary hardware and software information is collected for comparison with the actual state in the next step. See Figure 3 for a visual representation of this process.

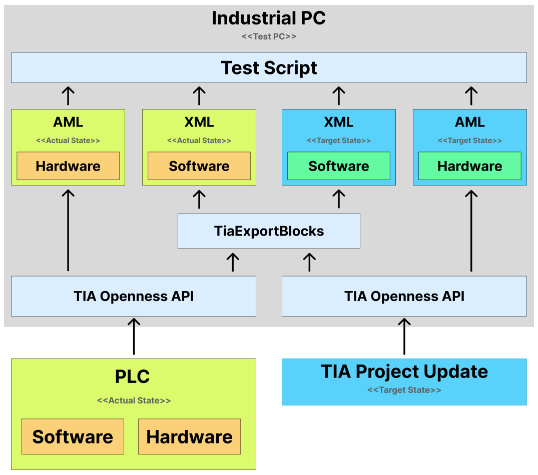

Figure 3. Determination of target and actual state during software update.

3.3. Determination of the Actual State

To determine the current state of software and hardware components, the network state is first assessed using the DCP Scan. Then, the I-PC programmatically contacts the PLC via the Siemens TIA-Portal Openness API [25].

The DCP "Identify All" command is initially broadcasted over the Ethernet network connected to the test PC. This command, illustrated in Figure 4, helps identify PROFINET devices physically connected to the network and supporting the DCP protocol. These devices, once found, return hardware information such as device name, IP address, firmware version, and MAC address via the DCP "GET" command.

Subsequently, the PLC software is downloaded to the I-PC through the TIA Openness API. Upon successful download, the AML file for the hardware configuration is generated by selecting "Export CAx data" from the "Tools" tab.

Additionally, variables in the PLC code can be automatically exported to machine-readable XML files from an open project using the open-source software "TiaExportBlocks."

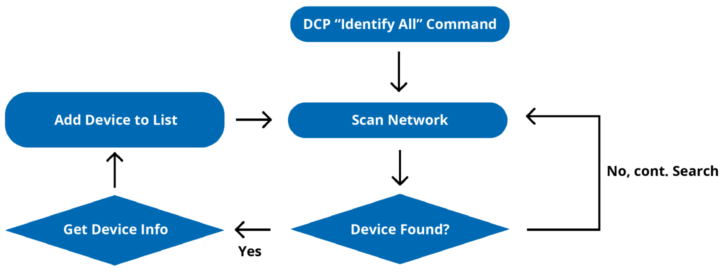

Figure 4. DCP command flow, based on [26].

3.4. Comparison of Target-Actual State

The core element of the concept is the comparison of the target and actual state, which emerges from the previous chapters. The components and IDs of the target software is compared to the actual software from the PLC and the imported expected hardware AML structure is compared to the determined DCP scan output. In doing so, the differences are identified and possible problems and inconsistencies are revealed.

When looking at the differences, incompatibilities between the hardware and software components can be identified and categorized in a table (see Chapter 3.5). The complete comparison process is automated by a custom test script on the I-PC. This automated test process can be performed when new software is uploaded or when the system is rebooted.

3.5. Error Detection Table

Shown below is the error detection table that is automatically created by the test script as a result of the compatibility check. The different rows are divided by components of the test flow, as shown in Figure 1. Possible reactions to the incompatibilities found are also included in the table. To simplify the notations within the tables, the following abbreviations are introduced:

- Hardware (**H/W**)
- Software (**S/W**)
- Not available (**n.a.**)
- Locally solvable problem (**L**)
- Remotely solvable problem (**R**)

Table 1. Error and inconsistency detection table.

Stage	Detectable error	Reaction
1 - PLC self test	CPU, E/A-Modul n.a. Error in the application program Data exchange faulty Memory integrity violated Comm. Interface n.a.	Repair hardware on site, install,replace(L) Check code and update(R) Check PLC(L) Check memory(L) Check power supply(L)
2 - Import & export of the target state	Error in the logic of the PLC code H/W topology n.a. SPS code n.a. File/XML structure incorrect	Fix logic errors in code(R) Check file structures(R) Check file structures(R) Re-export+import AML(R)
3 - Determination of the actual software state	PLC code n.a. PLC system n.a. PLC in wrong network	Check connection to PLC(L) Check connection to PLC(L) Check network configuration(R)
4 - Determination of the actual hardware state	No H/W devices found DCP Scan unsuccessful Data from device not retrievable H/W device in wrong network	Check connections(L) Check if DCP protocol is supported(L) Check connections(L) Check network configuration(R)

3.6. Reactions to Incompatibilities

The reactions outlined in chapter 3.5 offer a framework for test engineers to respond effectively to identified incompatibilities. It's important to note that these reactions are not automated by the system and require intervention from the test engineer.

The proposed concept for reacting to incompatibilities involves pinpointing at which location in the system action is needed to resolve the issues. Additionally, the error detection tables identifies whether a specific incompatibility can be fixed remotely (R) or if an on-site engineer intervention (L) is required.

4. Proof of concept

In the following, the TIA portal and other software tools from Siemens will be used explicitly to evaluate the strategy concept using a reference system to conduct a proof of concept. The evaluation method to be used is a test of the concept on a real plant.

4.1. Proof of concept criteria

The criteria for the evaluation correspond to the following requirements for the compatibility test:

A. The compatibility test must make a statement about the compatibility, i.e. the compatibility of the simultaneous operation of the hardware and software components connected to the PLC, and display this to the test engineer.

B. The compatibility check must be able to run automatically during the PLC restart or during the loading of software onto the PLC.
C. The result of the compatibility check must enable a statement to be made about incompatibilities and errors found.
D. At the end of the compatibility check, the test engineer should be shown appropriate reactions to incompatibilities and errors found.
E. The compatibility check should work with hardware and software from different manufacturers in the industry.

4.2. Reference system

The evaluation is carried out by testing the installation of new software on a hardware PLC on a reference system from the biopharmaceutical equipment supplier Sartorius. The concept presented for the compatibility check is applied and carried out. Two Sartorius industrial control systems are being used as possible real world application examples for the compatibility test. There are two different control system models which are used for the proof of concept.

One of the models contains a so-called software PLC. A virtual instance of a fully configurable and usable PLC is created on the I-PC. Communication with the field devices is then realized via an I/O module connected to the I-PC. Most engineering companies also uses automation scripts in its systems, which make it easier to process updates. WinCC OA is also used to implement a human-machine interface.

The second model uses a regular hardware PLC. This is supplied by an external power supply and connected to the I-PC via Ethernet to enable the PLC to be programmed. The hardware PLC is physically connected to the field devices via digital and analog I/O modules.

The assumed real world existing systems are developed by Sartorius with the TIA Portal from Siemens and are therefore suitable for demonstrating the application of the concept. However, no Sartorius control system could be used to evaluate the concept on site. Therefore, the individual steps of the compatibility test were evaluated on a system from the P2O-Lab [6] at TU Dresden. The industrial control systems in the P2O-Lab mostly correspond to the real world models with hardware PLCs.

5. Discussion

The compatibility check concept presented gives test engineers of industrial control systems an overview of incompatibilities before they occur during operation of the control system. Differences between the target and actual status are shown and possible reactions are identified.

By using an I-PC, the compatibility check is carried out automatically when the PLC is restarted or new software is installed. This means that errors and inconsistencies are detected at an early stage and can be rectified accordingly.

The concept facilitates an automated comparison of the hardware and software of a PLC's target and actual states, allowing the detection of inconsistencies or incompatibilities. This check can be applied when restarting the PLC or when updating the software. The error detection methods in chapter 3.5 offer an overview of potential errors and reactions. These tables are not complete and cover more inconsistencies in practice than presented, but have been condensed to focus on the core concept.

The concept relies solely on Siemens software and the TIA Portal. Future work could explore extending it to other major industrial control manufacturers, contributing to broader research in this domain.

Since the proposed concept does not include an own implementation of a test script, this could also be a useful extension to further verify the applicability of the presented concept. The script could be a starting point for a more in-depth analysis on how the concept could be integrated into other domains of industrial control systems, where different types of data sets are available.

6. Conclusion

In this work the need for automated compatibility testing is outlined and underpinned by a concept, based on a literature review. The findings of the literature review were presented in the chapter on the state of the art, existing test strategies from VIBN and CS were discussed and the concept of compatibility from various literature sources was brought together and examined. The SiL and HiL strategies from the VIBN and anomaly-based detection from the CS were considered to be particularly relevant and decisive for the compatibility testing presented in this study.

Based on this, the most important requirements for the concept were then derived and identified by means of a requirements analysis.The results of the requirement analysis are shown in the concept section of this work.

Furthermore, it was discussed how the strategies determined from the VIBN and the CS can be transferred to the processes of restarting the PLC and installing new software on the PLC.

The proposed concept for carrying out an automatic compatibility check was developed using applied research into the early detection of incompatibilities. A 4-phase concept was presented, which is characterized by the comparison of target and actual states of the software and hardware components.

As a result of the compatibility check, the concept offers an overview of incompatibilities found and shows possible reactions.

The functionality of the concept was implemented and evaluated on a module of the P2O Lab at TU Dresden. The results met the concept requirements and made it possible to identify incompatibilities.

7. Future Directions

There are numerous avenues for further research in the field of compatibility testing, building upon the theoretical basis established in this work. These possibilities include:

1. Implementation Variations: Expanding on the presented concept by creating various implementations to assess its flexibility and adaptability to different scenarios. Other scenarios could also involve completely different industry domains.
2. Multi vendor Compatibility: Evaluating and extending the concept to encompass software and hardware configurations from a range of manufacturers, providing a more comprehensive solution.
3. Automated Test Script: Developing a test script that automates the different phases of the concept, streamlining the compatibility testing process.
4. Data Sources Extension: Expanding the concept to incorporate data from additional sources for determining target and actual states, enhancing its robustness and applicability.
5. Fully Automated Systems: In the future, fully automated compatibility testing systems could significantly benefit test engineers and integrators of industrial control systems. This would enable early detection of incompatibilities in various Industry 4.0 components, particularly in the face of new hardware and software developments and changes to existing PLC architectures.

These research directions show the potential to advance the field of compatibility testing, making it a valuable asset in the ever-evolving landscape of industrial control systems.

Acknowledgments: We would like to acknowledge the support and insights, which our industry partner Sartorius has contributed to the proposed concept.

Conflicts of Interest: The authors declare no conflict of interest.

Abbreviations

The following abbreviations are used in this manuscript:

AML	Automation Markup Language
CAEX	Computer Aided Engineering Exchange
CS	Cybersecurity
DCP	Discovery and Configuration Protocol
DoS	Denial of Service
H/W	Hardware
HiL	Hardware-in-the-Loop
I-PC	Industrial Personal Computer
I/O	Input/Output
KNN	K next neighbors algorithm
PLC	Programmable Logic Controller
S/W	Software
SiL	Software-in-the-Loop
SVM	Support vector machines
TIA	Totally Integrated Automation (Software from Siemens)
VC	Virtual Commisionning
XML	Extensible Markup Language

References

1. Luder, A.; Klostermeyer, A.; Peschke, J.; Bratoukhine, A.; Sauter, T. Distributed automation: PABADIS versus HMS. *IEEE Transactions on Industrial Informatics* **2005**, *1*, 31–38. Conference Name: IEEE Transactions on Industrial Informatics, doi:10.1109/TII.2005.843825.
2. Mcfarlane, D.C.; Bussmann, S. Developments in holonic production planning and control. *Production Planning & Control* **2000**, *11*, 522–536. doi:10.1080/095372800414089.
3. Thramboulidis, K. The 3+1 SysML View-Model in Model Integrated Mechatronics. *JSEA* **2010**, *3*, 109–118. doi:10.4236/jsea.2010.32014.
4. Ulewicz, S.; Schütz, D.; Vogel-Heuser, B. Software changes in factory automation: Towards automatic change based regression testing. IECON 2014 - 40th Annual Conference of the IEEE Industrial Electronics Society, 2014, pp. 2617–2623. ISSN: 1553-572X, doi:10.1109/IECON.2014.7048875.
5. Zeller, A. *Absicherung von verteilten Automatisierungssystemen nach Änderungen der Steuerungssoftware*; Vol. 2019,2, *IAS-Forschungsberichte*, Shaker Verlag: Düren, 2019.
6. Vogt, L.; Pelzer, F.; Klose, A.; Khadyrov, V.; Lange, H.; Viedt, I.; Urbas, L.; Mädler, J. P2O-Lab: A Learning Factory for Digitalization and Modularization, 2023. doi:10.2139/ssrn.4456423.
7. Rothhaupt, M. Teststrategien für Software- und Hardwarekompatibilität in industriellen Steuerungen. https://nbn-resolving.org/urn:nbn:de:bsz:14-qucosa2-873589, 2023.
8. Hill, R.B.; Delbos, J.; Trebosc, S.; Tsague, J.; Feroldi, G.; Martin, J.; Master, T.; Lassabe, N. Improving interoperability of Virtual Commissioning toolchains by using OPC-UA-based technologies. 2021 26th IEEE International Conference on Emerging Technologies and Factory Automation (ETFA), 2021, pp. 01–07. doi:10.1109/ETFA45728.2021.9613490.
9. Muresan, M.; Pitica, D. Software in the Loop environment reliability for testing embedded code. 2012 IEEE 18th International Symposium for Design and Technology in Electronic Packaging (SIITME), 2012, pp. 325–328. doi:10.1109/SIITME.2012.6384402.
10. Mazza, L. Virtual commissioning of a pneumatic servosystem with a PLC in MiL, SiL, and HiL. https://webthesis.biblio.polito.it/11663/1/tesi.pdf, 2018.
11. Oppelt, M.; Wolf, G.; Urbas, L. Towards an integrated use of simulation within the life-cycle of a process plant. 2015 IEEE 20th Conference on Emerging Technologies & Factory Automation (ETFA), 2015, pp. 1–8. ISSN: 1946-0759, doi:10.1109/ETFA.2015.7301521.
12. Wu, D.; Ren, A.; Zhang, W.; Fan, F.; Liu, P.; Fu, X.; Terpenny, J. Cybersecurity for digital manufacturing. *Journal of Manufacturing Systems* **2018**, *48*, 3–12. doi:10.1016/j.jmsy.2018.03.006.
13. Ourston, D.; Matzner, S.; Stump, W.; Hopkins, B. Applications of hidden Markov models to detecting multi-stage network attacks. Proceedings of the 36th Annual Hawaii International Conference on System Sciences, 2003, pp. 10–19. doi:10.1109/HICSS.2003.1174909.

14. Mukkamala, S.; Janoski, G.; Sung, A. Intrusion detection using neural networks and support vector machines. Proceedings of the 2002 International Joint Conference on Neural Networks, 2002, Vol. 2, pp. 1702–1707.
15. Pan, Z.S.; Chen, S.C.; Hu, G.B.; Zhang, D.Q. Hybrid neural network and C4.5 for misuse detection. Proceedings of the 2003 International Conference on Machine Learning and Cybernetics, 2003, Vol. 4, pp. 2463–2467.
16. Zhang, J.; Liu, G.; Yu, W.; Ouyang, M. Adaptive control of the airflow of a PEM fuel cell system. *Journal of Power Sources* **2008**, *179*, 649–659. doi:10.1016/j.jpowsour.2008.01.015.
17. Gaddam, S.; Phoha, V.; Balagani, K. K-Means+ID3: A novel method for supervised anomaly detection by cascading k-Means clustering and ID3 decision tree learning methods. *IEEE Transactions on Knowledge and Data Engineering* **2007**, *19*, 345–354. doi:10.1109/TKDE.2007.44.
18. Liao, Y.; Vemuri, V.R. Use of K-Nearest Neighbor classifier for intrusion detection. *Computers & Security* **2002**, *21*, 439–448. doi:10.1016/S0167-4048(02)00514-X.
19. Sabhnani, M.; Serpen, G. Application of Machine Learning Algorithms to KDD Intrusion Detection Dataset within Misuse Detection Context. International Conference on Machine Learning; Models, Technologies and Applications, 2003, pp. 209–215.
20. Lee, K.; Kim, J.; Kwon, K.H.; Han, Y.; Kim, S. DDoS attack detection method using cluster analysis. *Expert Systems with Applications* **2008**, *34*, 1659–1665. doi:10.1016/j.eswa.2007.01.040.
21. Suteu, C. TiaExportBlocks: Tia Openness Export Blocks. https://github.com/Speedstuff/TiaExportBlocks, 2020.
22. AutomationML. What is AutomationML? – AutomationML. https://www.automationml.org/about-automationml/automationml/, 2022.
23. IEC. 62424 - Ed. 1.0. Representation of process control engineering – Requests in P&I diagrams and data exchange between P&ID tools and PCE-CAE tools. Technical report, IEC, 2016.
24. PLCopen foundation. XML Formats for IEC 61131-3. https://www.plcopen.org/system/files/downloads/tc6_xml_v201_technical_doc.pdf, 2009.
25. Siemens Openness Api Handbuch. Openness: API für die Automatisierung von Engineering-Workflows. https://cache.industry.siemens.com/dl/files/533/109798533/att_1069906/v1/TIAPortalOpennessdeDE_de-DE.pdf, 2021.
26. PROFINET University. DCP - Discovery and Configuration Protocol. https://profinetuniversity.com/naming-addressing/profinet-dcp/, 2018.