Alexandra Tóth

Intelligent Fault Detection During Commissioning of Industrial Automation Systems

A Hybrid and Explainable Approach

School of Technology and Innovations
Automation and Computing Science
Master's programme in Computing Sciences

Vaasa 2026

UNIVERSITY OF VAASA
School of Technology and Innovations
Author: Alexandra Tóth
Title of the thesis: Intelligent Fault Detection During Commissioning of Industrial Automation Systems: A Hybrid and Explainable Approach
Degree: Master of Science in Technology
Degree Programme: Master's degree programme in Computing Sciences
Supervisor: Timo Mantere
Year: 2026 **Pages:** 59

ABSTRACT:

Teollisuusautomaatiojärjestelmän käyttöönotossa on varmistettava, että logiikka, valvomojärjestelmä, logiikan ja valvomojärjestelmän välinen kommunikointi, Runtime-tietokanta ja historiankeruu vastaavat toisiaan. Poikkeama tässä ketjussa ei aina tarkoita prosessi- tai laitevikaa. Se voi johtua esimerkiksi keskeneräisestä konfiguraatiosta, puuttuvasta valvomomuuttujasta, väärästä tietotyypistä, heikosta kommunikoinnin laadusta tai siitä, ettei historiankeruu ole vielä tuottanut näytteitä.

Tässä diplomityössä kehitettiin käyttöönottoa tukeva analyysimenetelmä teollisuusautomaatiojärjestelmien signaali-, konfiguraatio- ja logiikkatason poikkeamien havaitsemiseen. Menetelmä yhdistää sääntöpohjaisen analyysin ja päätöspuupohjaisen koneoppimismallin. Sääntöpohjainen osa tarkastaa suoraan todennettavia asioita, kuten valvomomuuttujien löytymistä, tietotyyppien vastaavuutta, kommunikoinnin laatua, ajonaikaisia arvoja ja historiankeruun näytteitä. Päätöspuuta käytetään tukevana vertaisvertailuna, jossa samankaltaisia entiteettejä verrataan toisiinsa rakenne- ja konfiguraatiotason poikkeamien tunnistamisen tukemiseksi. Menetelmää arvioitiin esikäyttöönotossa ja käyttöönotossa. Esikäyttöönotossa logiikkaa simuloitiin ohjelmointiympäristön simulointiominaisuudella, jolloin analyysia voitiin käyttää konfiguraation ja signaaliketjun tarkastamiseen ennen varsinaista käyttöönottoa. Käyttöönotossa signaalit luettiin laitoksen oikealta logiikalta, jolloin analyysi pystyi hyödyntämään todellisia ajonaikaisia arvoja ja historiankeruun näytteitä.

Tulosten perusteella sääntöpohjainen analyysi soveltuu hyvin käyttöönoton selkeiden ja todennettavien poikkeamien tunnistamiseen. Päätöspuuhun perustuva vertaisvertailu tuo lisätukea silloin, kun järjestelmässä on riittävästi samankaltaisia entiteettejä vertailua varten. Simuloidut entiteettitason koulutustapaukset soveltuvat päätöspuumallin kehittämiseen ja rakenteen tarkastamiseen, mutta ne eivät yksin riitä osoittamaan mallin luotettavuutta todellisessa käyttöönotossa. Työn tuloksena syntyi menetelmä, joka kokoaa havainnot, niiden perusteet, entiteettikohtaisen taustan, selitykset ja mahdollisen vertaisvertailun yhteen insinööriraporttiin. Menetelmä ei tee lopullisia käyttöönottopäätöksiä eikä korvaa käyttöönottoinsinöörin arviointia. Sen tarkoitus on auttaa kohdistamaan tarkastus niihin kohtiin, joissa automaatiojärjestelmän eri kerrokset eivät vastaa odotettua kokonaisuutta.

KEYWORDS: Industrial Automation, Commissioning, Fault Detection, Machine Learning, Decision trees, Rule-Based Systems, Hybrid Methods, Explainability

Contents

Figures

Tables

Abbreviations

AI	Artificial Intelligence
API	Application Programming Interface
DDT	Derived Data Type
FDD	Fault Detection and Diagnosis
HTML	HyperText Markup Language
ISO/IEC/IEEE	International Organization for Standardization/International Electrotechnical Commission/Institute of Electrical and Electronics Engineers
JSON	JavaScript Object Notation
ML	Machine Learning
OPC	Open Platform Communications
OPC UA	Open Platform Communications Unified Architecture
PCA	Principal Component Analysis
PLC	Programmable Logic Controller
PLS	Partial Least Squares
SCADA	Supervisory Control and Data Acquisition
SQL	Structured Query Language
XAI	Explainable Artificial Intelligence

1 Introduction

An industrial automation system typically comprises field devices such as sensors and pumps, programmable logic controllers (PLC), supervisory systems (SCADA), communication layers, and data management systems. This architecture is complex and tightly integrated, meaning that errors may compromise operational reliability and safety. As a result, before taking such a system into use, it must be tested thoroughly. The testing usually includes many levels, including end-to-end validation and system-level consistency. Commissioning commonly involves extensive manual validation of signal mappings and integration consistency. Unlike equipment malfunctions, issues during commissioning frequently result from inconsistencies in configuration, communication, or signal mapping. To examine this problem more closely, it is necessary to review existing approaches to fault detection and diagnosis in industrial automation systems.

1.1 Commissioning in Industrial Automation

Commissioning takes place between installation and full operational use, and it represents the final validation phase of an automation project. Schamp et al. (2020) describe commissioning as typically being performed on the real hardware, meaning that verification takes place in the actual production environment rather than in a simulated test environment.

During commissioning, the automation system is tested end-to-end to verify that data flows correctly from field devices to the supervisory level and back. Signals must be validated throughout the entire control architecture to ensure that measurements, logic, and control actions operate as required. According to Schamp et al. (2020), correct system behaviour and early testing are essential to reduce risks during plant operation.

Modern automation systems include a large number of signals and dependencies across system layers. Configuration changes are often introduced late in the project phase, and integration issues may only become visible when subsystems are connected. Commissioning is often conducted in difficult circumstances, including delays and time pressure. As Schamp et al. (2020) state “In a traditional workflow, the main commissioning part of industrial control systems is performed on the real set-up and consequently during a time critical phase of the project.”

Validation is manual and commonly based on checklists and signal-by-signal testing. Increasing system complexity adds to the workload, often resulting in extensive manual debugging efforts (Schamp et al., 2020). Because of the manual nature of the process, errors remain undetected due to oversight or limited testing time. Given the scale and complexity of commissioning activities, additional systematic diagnostic support could improve reliability and efficiency. This creates a clear motivation to examine how fault detection and analysis methods can support commissioning engineers during this phase.

1.2 Fault Detection in Industrial Automation

Continuous supervision is essential to ensure safe and reliable operation of an industrial automation system. In automatic control, supervisory functions are responsible for identifying undesired or unpermitted process states and initiating appropriate counteractions (Isermann, 1997). Fault detection and diagnosis (FDD) form a central part of supervision. Fault detection refers to the identification of deviations from normal operation. Fault diagnosis aims to determine the type, location, and possible size of the fault (Isermann, 2005). In industrial automation systems, it is vital to identify faults before they lead to malfunctions or system failures. The objective of early fault detection is to provide sufficient time for corrective actions such as reconfiguration, maintenance, or repair (Isermann, 1997).

Various methods have been developed for fault detection and diagnosis in industrial automation systems. These methods include model-based approaches and data-driven techniques. Model-based methods use mathematical models of the system, whereas data-driven methods use measured process data to identify abnormal behaviour. Both approaches are widely used in industrial applications.

Most research focuses on fault detection during normal system operation. In many cases, the system is assumed to operate in a stable production phase. However, fault detection challenges also appear during commissioning, when the system configuration is still being verified, and different subsystems are being integrated. Traditional fault detection approaches are not always designed for these conditions.

1.3 Research Gap

Despite the availability of various fault detection methods, explainable diagnostic methods designed specifically for commissioning are not clearly addressed in the literature. Some approaches aim to support commissioning through virtual commissioning or digital twin technologies. These methods utilise simulation models to test control logic before physical deployment. While such approaches can reduce risks, they depend on accurate virtual models and do not remove the need for validation in the real system environment.

Commissioning engineers often work with actual configuration files, database structures, and runtime data. Errors may relate to incorrect signal mappings, inconsistent data types, or integration mismatches between subsystems. The use of diagnostic methods directly on real configuration and runtime data during commissioning is less visible in the literature.

1.4 Research Questions

Based on the identified research gap, this study investigates the role of a hybrid and explainable diagnostic approach during commissioning.

Research Question 1: How can a hybrid, explainable analysis agent combining rule-based logic and machine learning support fault detection during commissioning?
Research Question 2: To what extent can simulated entity-level training cases be used to support the training and model structural validation of the machine learning component during the development phase?
Research Question 3: How does a hybrid and explainable analysis approach support commissioning engineers in interpreting signal-level and logical anomalies?

The following chapters present the theoretical background, system design, and validation of the proposed approach.

1.5 Scope and Delimitations

The scope of this study is limited to configuration mismatches and integration-related faults that occur during the commissioning phase of industrial automation systems. The focus is on errors related to signal mappings, data types, and inconsistencies between interconnected subsystems such as PLC, SCADA, communication, and data management layers. The proposed approach is designed to support commissioning engineers in identifying these types of faults using real project configuration and runtime data.

This study does not address mechanical or electrical component failures, nor does it focus on process-level physical faults that may occur during long-term system operation. It neither aims to develop new machine learning algorithms nor digital twin models, and

cybersecurity aspects as well as large-scale production monitoring remain outside the scope of this study.

2 Literature Review

The purpose of this chapter is to review the main research directions related to fault detection and diagnosis. The literature includes model-based approaches, process-level diagnostic strategies, and data-driven monitoring techniques. In some cases, different methods are combined. As machine learning is applied in industrial contexts, issues such as dataset shift and model transparency have become relevant. This chapter presents the theoretical foundation that supports the development of a hybrid and explainable diagnostic approach for commissioning.

2.1 Model-Based Fault Detection

Model-based fault detection is based on the use of mathematical or physical models that describe how a system is expected to behave. The main principle is simple: measured process variables are compared with values calculated by the model. If the difference between the measured and calculated values exceeds the predefined limit, the system is considered to deviate from the normal operation.

Isermann (1997) explains that model-based methods rely on analytical redundancy. Instead of using additional hardware sensors, the process model is used to generate residuals. A residual represents the difference between measured and estimated values. When the difference becomes large enough, it may indicate a fault. Fault diagnosis then aims to determine the type and location of the deviation.

Venkatasubramanian et al. (2003a) describe these methods as quantitative model-based techniques. In this approach, explicit mathematical models derived from first principles or system identification are used for fault detection and isolation. When accurate models

are available, this approach can provide structured information about the system behaviour and support systematic analysis.

However, model-based methods depend strongly on the quality of the underlying model. In complex industrial systems, obtaining accurate models can be difficult because of nonlinear behaviour, parameter uncertainties, and changing operating conditions. If the model does not represent the actual system correctly, residuals may indicate deviations that are not related to real faults.

This becomes challenging during commissioning. At that stage, system configuration may still change; parameters may not yet be fully tuned, and integration between subsystems may be incomplete. In such conditions, deviations detected by the model may result from configuration inconsistencies rather than physical faults. For this reason, purely model-based approaches may have limitations when applied directly during commissioning.

2.2 Qualitative and History-Based Fault Detection

Process-level fault detection focuses on the overall behaviour of the process rather than on detailed mathematical models of individual components. Instead of comparing measured values to model-based predictions, these methods analyse patterns, trends, and relationships between process variables.

Venkatasubramanian et al. (2003b) describe qualitative model-based methods as approaches that use structural knowledge of the process, such as cause-effect relationships and logical connections between variables. These methods do not always require precise numerical models. Instead, they rely on qualitative reasoning or rule-based structures to identify abnormal situations. This can make them more flexible in complex systems where accurate quantitative models are difficult to obtain.

In addition to qualitative approaches, history-based methods use past process data to detect deviations. Venkatasubramanian et al. (2003c) classify these as process history-based methods, which rely on historical process data for feature extraction. The quantitative approaches within this category formulate fault diagnosis as a pattern recognition problem.

Process-level methods can be useful in large-scale industrial systems because they consider interactions between multiple variables. They may detect faults that are not visible at the component level. However, their effectiveness depends on the availability of representative process data and clearly defined normal operating conditions.

During commissioning, defining normal operation can be challenging. The system may not yet operate in a stable production state, and configuration changes may still occur. As a result, historical references or qualitative assumptions may not fully represent the actual system behaviour. This can reduce the reliability of purely process-level approaches during commissioning.

2.3 Data-Driven Industrial Monitoring

Data-driven fault detection differs from model-based approaches in that it does not require prior physical and mathematical knowledge of the process. Shen et al. (2014) state that model-based fault detection requires prior knowledge of the process. Data-driven approaches instead analyse large sets of recorded process data. In such methods, the necessary process information is derived directly from data rather than from explicit process models.

A central category of data-driven methods is multivariate statistical process monitoring. Shen et al. (2014) identify principal component analysis (PCA) and partial least squares (PLS) as widely applied techniques in industrial process monitoring. These methods

project correlated process variables into a lower-dimensional subspace and apply statistical test measures for fault detection. In this way, deviations are detected through statistical evaluation rather than through consistency checks against first-principle models.

Data-driven methods are relevant in complex industrial equipment where accurate mathematical modelling is difficult. Sahu et al. (2024) emphasise that industrial systems are structurally complex and that their behaviour is difficult to model due to internal dependencies and environmental interactions. In such cases, data-driven approaches combined with expert knowledge are often preferred.

At the same time, data-driven monitoring introduces challenges. Blázquez-García et al. (2021) note that observations identified as outliers in time series may correspond to noise, errors, or otherwise unwanted data. This complicates anomaly detection because such observations may not represent meaningful system behaviour. The authors also discuss methods designed for data streams where underlying distribution may change over time, referring to situations where data can be subject to concept drift. These characteristics make the construction of stable anomaly detection models challenging.

During commissioning, stable and representative historical data may not yet be available, and system configurations may still change. Under such conditions, the assumptions required for data-driven monitoring may not fully hold. This limits the applicability of purely data-driven approaches during the commissioning phase.

2.4 Hybrid Fault Diagnosis Approaches

Hybrid fault diagnosis approaches combine multiple diagnostic methods within a monitoring framework. In industrial automation systems, individual fault detection methods often have limitations when used independently. For this reason, research has explored diagnostic systems that integrate different types of knowledge and analysis methods.

Venkatasubramanian et al. (2003a, 2003c) classify fault detection and diagnosis approaches into several categories, including quantitative model-based methods, qualitative model-based reasoning, and process history-based methods. While these categories are often studied separately, practical diagnostic systems may benefit from combining information from several sources. Such integration can improve diagnostic performance when the assumptions of a single method are not fully satisfied.

In industrial monitoring applications, hybrid approaches frequently combine data-driven analysis with engineering knowledge or rule-based reasoning. Shen et al. (2014) note that statistical monitoring techniques are often used together with process knowledge to improve fault detection and interpretation. In such systems, statistical models may detect abnormal patterns in process data, while expert knowledge helps interpret the meaning of these deviations.

Recent research also highlights the role of machine learning in hybrid diagnostic systems. Sahu et al. (2024) describe industrial diagnostic frameworks where machine learning techniques are applied together with domain knowledge to analyse complex industrial equipment. These approaches attempt to combine the pattern recognition capability of machine learning with the interpretability provided by engineering knowledge.

A hybrid diagnostic system can therefore provide a more flexible framework for fault detection and diagnosis in complex industrial environments. At the same time, integrating different diagnostic components requires careful system design and clear interpretation of the results. Without such structure, hybrid systems may become difficult to analyse or validate (Venkatasubramanian et al., 2003c; Sahu et al., 2024). This motivates the use of a hybrid approach in commissioning conditions, where no single method can fully address configuration-related inconsistencies.

2.5 Decision Tree-Based Machine Learning

Machine learning methods are increasingly applied in industrial monitoring and fault diagnosis. Data-driven approaches can analyse large amounts of operational data and identify patterns that may indicate abnormal system behaviour (Sahu et al., 2024; Hamrouni et al., 2023). In these approaches, models learn relationships between variables directly from data rather than relying on explicit physical models.

Decision trees are one of the commonly used machine learning methods for classification and decision support. Braga-Neto (2024) describes decision trees as models that partition the feature space into regions associated with different class labels. The resulting structure can be represented as a hierarchical tree where each node corresponds to a decision rule based on feature values.

In diagnostic applications, decision trees are attractive because their reasoning process can be expressed in a transparent rule-based form. Each branch represents a decision condition, and each leaf node corresponds to a predicted class or diagnostic result. This structure allows engineers to follow how a conclusion has been reached.

Interpretability is an important consideration when machine learning methods are applied in technical systems. Arp et al. (2020) emphasise that machine learning models should be carefully used in engineering contexts and that model behaviour should be understandable to system operators. When models are used for decision support, transparent reasoning can improve trust and allow experts to verify the results.

Practical deployment of machine learning models can also introduce challenges. Sculley et al. (2015) discuss how machine learning systems may accumulate hidden technical complexity when models interact with real operational data. These challenges highlight the importance of selecting methods that remain understandable and manageable in engineering environments.

For these reasons, decision tree models are often considered suitable for diagnostic support tasks where both predictive capability and interpretability are required. In this study, decision tree-based analysis is used as a component of a hybrid approach designed to support commissioning engineers. In this work, decision tree analysis is not used as a standalone fault classifier, but as an interpretable supporting method within a hybrid analysis process.

2.6 Dataset Shift and Non-Stationarity

Machine learning models typically assume that the statistical properties of the training data remain similar during model deployment. When this assumption does not hold, the data encountered during operation may differ from the data used during training. This phenomenon is commonly referred to as dataset shift. Moreno-Torres et al. (2012) describe dataset shift as a situation where the joint distribution of input variables and target variables changes between the training and the application phases.

Dataset shift can appear in several forms depending on how the data distribution changes. Moreno-Torres et al. (2012) describe several forms of dataset shift. These include covariate shift, where the distribution of input variables changes, and prior probability shift, where the class distribution changes between training and application phases. In such situations, models trained on historical data may produce unreliable predictions because the learned relationships no longer represent the current environment.

In time-dependent systems, these changes are often described using the concepts of concept drift or non-stationarity. Lu et al. (2019) explain that concept drift occurs when the statistical relationship between input data and the target concept evolves over time. As a result, models that were previously accurate may lose predictive performance when the underlying system behaviour changes.

Industrial monitoring environments are sensitive to such changes because operating conditions and system configurations may evolve over time. Shen et al. (2014) note that industrial processes often generate large volumes of operational data under varying operating conditions. These variations can introduce challenges for data-driven monitoring systems because models trained under one set of conditions may not generalize well to other operating states.

Similar challenges have also been discussed in the context of machine learning systems deployed in real operational environments. Sculley et al. (2015) emphasise that machine learning models may degrade in performance when the data distribution encountered during operation differs from the training data. Such effects can introduce hidden technical challenges in practical machine learning deployments.

During commissioning, these issues may become even more visible. The automation system is still under configuration, and subsystems are still being integrated. Signal mappings, parameter settings, and communication structures may change during this phase. In addition, the data available during early system testing may not represent the behaviour of the fully configured system. These characteristics introduce dataset shift between development, testing, and operational phases, which complicates the application of purely data-driven diagnostic methods during commissioning.

2.7 Explainable Artificial Intelligence

Explainable Artificial Intelligence (XAI) refers to methods that make behaviour and decisions of machine learning models understandable to human users. Chamola et al. (2023) describe XAI as an approach that improves transparency and trust in artificial intelligence systems by providing explanations for model predictions.

Explainability is essential in industrial diagnostic applications, where machine learning results often support engineering decision-making. When models are used for fault detection or condition monitoring, engineers must be able to interpret the reasoning behind a diagnostic result. Brusa et al. (2023) note that explainable machine learning models can help engineers understand how different features contribute to fault classification and diagnostic outcomes.

Tree-based machine learning models are often considered more interpretable than many other machine learning approaches. Methods have been developed to analyse how individual input features influence predictions produced by tree models and to provide both local explanations for individual decisions and global insights into model behaviour (Lundberg et al., 2020). In engineering applications, interpretability is also important for ensuring that machine learning systems remain understandable and controllable by system developers and users (Arp et al., 2020).

In this study, decision tree-based machine learning is used as a supporting analysis method. The primary diagnostic reasoning and explanations are produced using rule-based analysis, while the machine learning component provides additional insight for identifying signal-level and logical anomalies during commissioning.

3 Methodology

The purpose of this chapter is to describe the methodological approach used to develop the proposed commissioning support tool. The approach is based on a hybrid analysis concept that combines rule-based logic and machine learning to detect signal-level and logical anomalies. The methodology follows a layered approach, where each layer enriches the dataset without discarding information. This design supports explainability, traceability, and robustness under incomplete commissioning data conditions.

Commissioning support in this work is treated as a system-level testing and validation support activity. The purpose of the tool is not to automate final acceptance decisions, but to collect evidence, identify deviations, and report observations in a form that supports engineering judgement. This is consistent with the view of software and systems testing, where testing is based on defined test items, test basis, observed results, and reporting of findings (ISO/IEC/IEEE, 2022).

The chapter first defines the commissioning context and the available data, followed by the key requirements for the tool. After this, the structure of the solution is described, focusing on how data is processed, analysed, and interpreted. Finally, the integration of the analysis results into a unified commissioning report is presented.

3.1 Commissioning Context and Data Sources

This study focuses on data that is available during commissioning without relying on historical process datasets. The analysis is based on configuration data and current runtime values obtained directly from the automation system.

The data used in this work is obtained from three sources that represent different levels of the system. PLC configuration files provide the definition of control logic, data types, and signal structure. This data describes how the system is intended to operate and is used as a reference in the analysis (PLCopen, 2016). At system level, the same signals and structures are represented in engineering and runtime databases. These databases contain signal definitions, object structures, and relationships between components, forming a structured representation of the configured system. In practice, this information is accessed using SQL queries.

Runtime data is obtained through OPC-based communication interfaces, which enable data exchange between devices and higher-level systems (Velesaca et al., 2025). This data represents the current state of the system during operation and provides the observable behaviour of signals.

The methodology is based on analysing consistency between these data sources. PLC configuration defines the intended behaviour, database structures represent the configured system, and OPC data reflects runtime behaviour. Fault detection is formulated as the identification of mismatches between these representations. Table 1 shows data sources and their role in the analysis.

Table 1. Data sources and their role in the analysis.

Source	Purpose in analysis	Example information
PLC configuration file	Defines intended control structure and signal model	DDTs, signal names, data types, bit mappings
OPC UA	Provides live runtime visibility	current values, timestamps, signal quality
Galaxy SQL database	Provides engineering model structure	objects, templates, attribute bindings
Runtime SQL database	Provides runtime and historical evidence	snapshots, historical values, availability

3.2 Requirements for Commissioning Support Tool

The commissioning support tool was designed to operate on configuration and runtime data available during system commissioning. The analysis does not rely on historical datasets or predefined system models but is based on data collected directly from the automation system.

The analysis was implemented as a read-only process. No modifications were made to PLC configuration, databases, or runtime data. All data processing was performed locally after collection. This approach ensures that the tool can be used during commissioning without affecting system operation.

Direct integration with SCADA user interfaces or vendor-specific APIs was not used. Instead, SQL-based access was selected to retrieve system configuration and structural information. This allows the analysis to be performed independently of the SCADA application layer while still providing access to relevant system data.

The analysis is performed at signal level. Each signal is processed individually using its available attributes, including configuration data, database information, and runtime values. The method compares these representations to identify inconsistencies between them. Missing or incomplete data is handled explicitly and is not treated as a fault without sufficient evidence.

The output of the tool is a structured dataset that combines all collected information for each signal. This dataset is used as the input for further analysis stages described in the following sections. The structure of the tool is described in Figure 1.

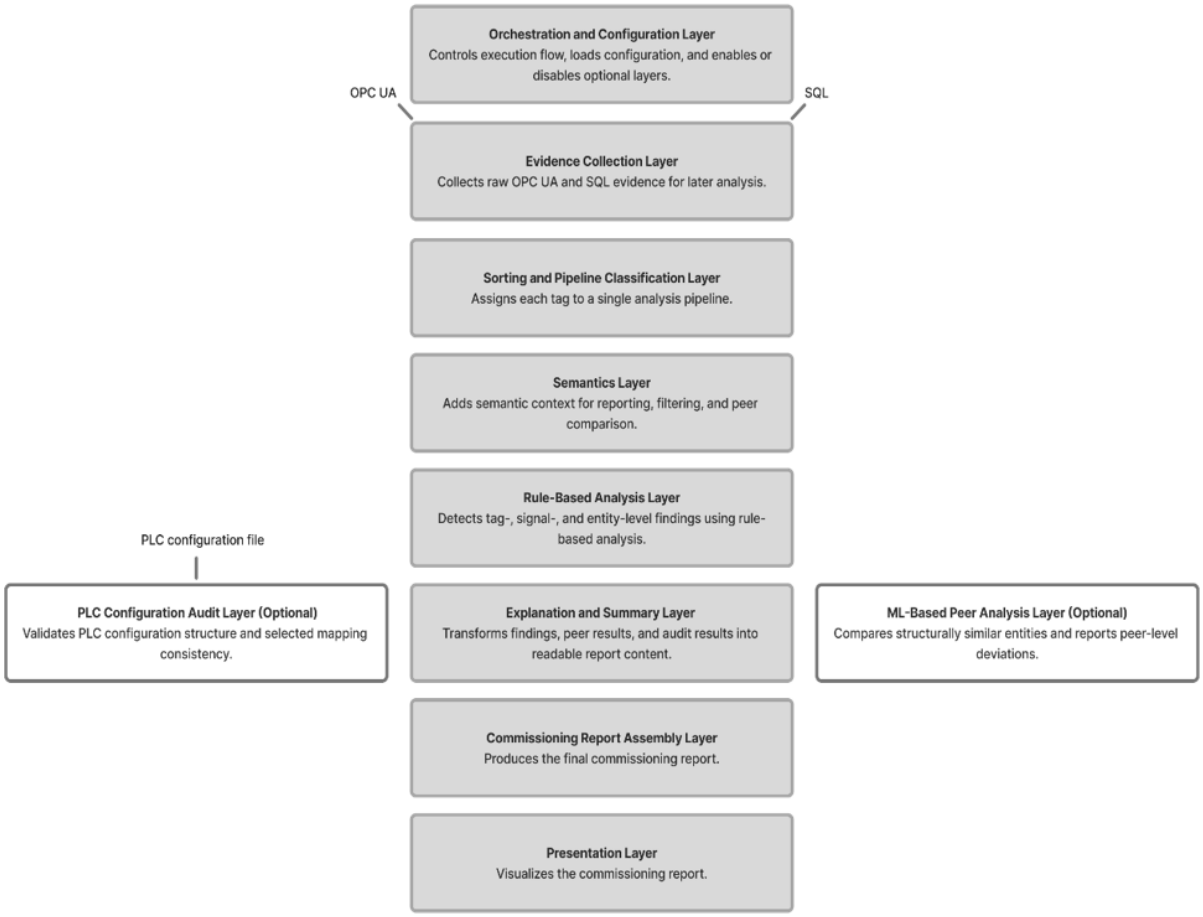

Figure 1. Commissioning support tool structure.

The commissioning support tool was developed in Python, and the data is stored in JSON format. Python was selected to support local execution, straightforward data processing, and integration of rule-based logic and lightweight machine learning components.

3.3 Evidence Layer

After data collection, the analysis proceeds by constructing a unified dataset from the available sources. This step is referred to as the evidence layer. Its purpose is to combine PLC configuration, SQL data, and OPC runtime data into a single structure that can be used in further analysis.

In this work, PLC configuration, OPC data, and SQL data are first read separately. PLC configuration provides the reference structure and signal definitions when available. OPC data provides the current runtime view of signals. SQL queries provide system-level information, including signal bindings, object structures, and metadata stored in the database. These sources are not assumed to be complete or consistent.

The collected data is merged at tag level. Each tag is represented as a single entry in the resulting dataset. All available attributes from PLC, SQL, and OPC are attached to the same tag entry. If a tag exists in one source, but is missing from another, this is preserved in the data. No attempt is made to correct or infer missing information at this point.

During the merge, initial flags are added to describe the state of the data. These flags indicate conditions such as missing OPC data, missing SQL mappings, or unavailable history data. The purpose of these flags is to record what was observed during data collection, not to classify faults. The same information is later used in the analysis without re-reading the original data sources.

After merging and flagging, all tag-level entries are written into a single JSON-based structure. This structure contains the combined data from all sources together with the generated flags. The structure is not filtered, and no tags are removed, even if the data is incomplete or inconsistent.

The evidence layer therefore produces a complete representation of the collected system data. This representation also preserves structural relationships between tags, which are later used in entity-level analysis.

3.4 Sorting Layer

After the evidence dataset has been constructed, the next step is to assign each tag to a processing pipeline. This step is referred to as sorting. Its purpose is to classify tags based on their data type and structure so that they can be analysed using appropriate logic in later stages.

Each tag is assigned to a pipeline. In this work, the pipelines are Boolean, Integer, FloatDouble, Word, and Array. The assignment is based on the attributes available in the evidence layer, such as data type, value format, and structural characteristics. However, project configuration is also used where the value type alone is not sufficient. Word tags are identified using configured exact names, suffixes, and regular expressions. These rules are used to route tags to the correct analysis pipeline, but they do not create diagnostic conclusions by themselves.

The sorting step does not perform any analysis or fault detection. It only determines how each tag will be processed in the rule-based layer. The output of this stage is the evidence dataset extended with pipeline information for each tag. This information is used in the next stage to route tags to the corresponding analyzers.

3.5 Semantics Layer

After sorting, the dataset is enriched with semantic information. The purpose of this step is to provide context for each tag without affecting the rule-based analysis logic.

The semantic roles are assigned to tags based on naming patterns and available metadata. These roles describe the purpose of the signal, such as measurement, status, parameter, or selector. The role assignment is based on consistent naming rules and does not depend on runtime behaviour.

Semantic information is not used for fault detection, and it does not affect the rule-based decisions, but it provides context that can be used in later stages. The result of this step is an enriched dataset where each tag includes both structural context and its semantic role. Semantic roles are added to the same dataset as the previous layer's information. Semantic information may later be used for reporting and peer-level comparison.

3.6 Rule-Based Analysis Layer

After the evidence dataset has been constructed and enriched with pipeline and semantic information, the analysis proceeds by applying rule-based checks to the data. The purpose of this stage is to evaluate each tag and entity based on the available information and to identify inconsistencies between configuration, database, and runtime data.

The analysis is performed separately for each pipeline. Tags have already been assigned to pipelines in the sorting layer, and each pipeline is processed using its own analyzer.

The analyzers use the attributes collected in the evidence layer together with pipeline information and semantic context. This includes data type, runtime value, availability of OPC data, database information, and the structural context provided by entities.

The rule-based analysis evaluates the consistency of each tag using explicit conditions derived from the available data. Typical checks include whether a tag exists in all required sources, whether the data type matches configuration and runtime, and whether a runtime value is available when expected. The analysis also uses the flags generated in the evidence layer to determine how each tag should be handled.

In addition to tag-level checks, the analysis is applied at entity level. Tags that belong to the same entity are evaluated together to detect inconsistencies within structured objects. This allows the analysis to identify issues that are not visible at individual tag level, such as missing signals or structural mismatches inside an entity.

The results are divided into findings and journal entries. Findings represent the actual analysis output and include both state information and detected issues. Journal entries are used to record analysis context, such as missing data, skipped checks or situations where a reliable conclusion cannot be made.

The rule-based layer does not use statistical models or pattern recognition. All results are based on checks that can be directly traced to the underlying data. Each finding is linked to the corresponding tag or entity and to the specific condition that triggered it. The output of the rule-based layer is an updated dataset where each tag and entity includes the results of the analysis.

3.7 Peer-Level Machine Learning Layer

In addition to rule-based analysis, a machine learning component is used to support the interpretation of results at entity level. Entity-level consistency is already evaluated in the rule-based layer, where tags are grouped based on structures derived from PLC configuration. The machine learning layer uses the same entity structure and operates on top of the existing analysis results without modifying them.

The machine learning implementation is based on a decision tree agent operating on entity-level feature vectors. Decision trees were selected because the results must remain explainable in a commissioning context. The model logic can be inspected directly from its structure, which allows the engineer to verify how a conclusion has been formed. In addition, the solution must remain lightweight and suitable for industrial environments. Decision trees can be executed with low computational overhead and do not require complex infrastructure or large-scale training.

More complex machine learning methods were not selected because explainability was a mandatory requirement in this work. The machine learning component had to support commissioning engineers by producing results that can be inspected, traced, and explained together with the rule-based findings. Methods such as neural networks, support vector machines, and ensemble models can be useful in classification tasks, but their decision logic is less directly visible to the engineer. Since the purpose of the machine learning layer was not to maximise classification performance, but to provide explainable peer-level support, a single decision tree was selected. Its structure can be inspected directly and its decision logic can be presented in a rule-like form.

An example of the decision tree inspection output is shown in Figure 2. The output shows the feature-based split conditions and feature importance values used by the trained peer-level model. The class label belongs to the synthetic training setup, and it is not interpreted as an independent fault conclusion in the commissioning report.

```
=== DECISION TREE (TEXT) ===

|--- history_changed_ratio <= 0.1503
|   |--- class: 1
|--- history_changed_ratio >  0.1503
|   |--- cfg_delay_sum_median <= 119.4118
|   |   |--- n_opc_status_bad <= 1.5000
|   |   |   |--- word_history_coverage_ratio <= 0.0517
|   |   |   |   |--- class: 1
|   |   |   |--- word_history_coverage_ratio >  0.0517
|   |   |   |   |--- class: 0
|   |   |--- n_opc_status_bad >  1.5000
|   |   |   |--- class: 1
|   |--- cfg_delay_sum_median >  119.4118
|   |   |--- class: 1


=== FEATURE IMPORTANCE ===

history_changed_ratio                    0.3444
cfg_delay_sum_median                     0.3154
n_opc_status_bad                         0.1823
word_history_coverage_ratio              0.1579
```

Figure 2. Example inspection output of the decision tree used in the peer-level machine learning layer.

The purpose of the machine learning layer is not to detect faults independently, but to provide a peer-level comparison between structurally comparable entities. For each entity, a peer summary is generated by comparing its signals to other similar entities. This allows the analysis to highlight peer-level differences that are not directly visible from individual tag checks. Table 2 summarises the conceptual differences between rule-based analysis and ML peer-level comparison in this work.

Table 2. Rule-based analysis vs. ML peer comparison.

Aspect	Rule-based analysis	ML peer comparison
Main role	Detect directly verifiable inconsistencies	Highlight relative differences between peers
Explainability	High	High
Output	Findings with reasoning	Supporting observations
Dependency on history	Low	Moderate
Dependency on peer population	None	Required
Used for severity classification	Yes	No

Peer groups are formed from structurally comparable entities. In this work, comparability is based on shared entity type, similar signal composition, pipeline distribution, and semantic roles where applicable.

For example, if one entity differs from others of the same peer group while the rest behave consistently, this may indicate a local issue rather than a system-wide problem. Conversely, if multiple entities show similar deviations, the issue may relate to shared configuration or communication. This type of comparison complements the rule-based findings and provides additional context for interpretation. If no comparable peers are available, the entity is treated as a singleton, and no peer-based conclusion is made.

During development, simulated entity-level training cases were used to train and initially test the decision tree component under controlled conditions. While such training cases can provide controlled peer-level differences, the selected approach relies on peer-level comparison of real commissioning data, where entities are evaluated relative to one another. This reduces the need for predefined fault cases and keeps the analysis aligned with the actual structure and variation of the commissioned system.

The output of this stage is a peer summary for each entity. The summary highlights deviations in entity-level feature values and structural attributes relative to structurally comparable peer entities. These results are not interpreted as independent fault conclusions, but as supporting information that is combined with rule-based findings in the following stages.

3.8 Explanation and Summary Layer

After rule-based and peer-level analysis, the results are further processed to support interpretation during commissioning. This stage focuses on explaining the findings and prioritizing them so that the engineer can identify the most relevant issues.

The analysis produces findings at both tag and entity level. These findings are extended with explanations that describe why a specific result was produced. The explanations are generated directly from the rule-based analysis logic and the conditions that triggered each finding. No additional machine learning or external models are used in this stage. Each explanation can be traced back to specific checks, flags, and data values in the evidence layer.

Explanations are used to make the results understandable in practical troubleshooting situations. For example, if a single tag is not working while other tags of the system are, it is not likely that there would be communication or a system-wide problem. At the entity level, the results provide a summary of the condition of a structured object, which helps to narrow down the location of the issue.

In addition to explanation, severity-based prioritisation is applied to the findings. The purpose of prioritisation is to make the most relevant commissioning issues visible first. Findings are normalized into engineering severity levels, such as high, medium, and info.

The report then orders findings according to these severity levels instead of calculating a universal risk score.

Prioritisation is necessary because commissioning often produces many findings. A finding that prevents reliable signal interpretation is more important than several informational observations. For this reason, tag-level issues are ordered by severity, each tag inherits its top severity from its most severe issue, and each entity inherits its top severity from its most severe tag. The machine learning component remains support-only information and does not override or escalate rule-based findings.

The result of this stage is a structured output where findings are explained and ordered according to their importance. This makes the analysis results usable in practice by helping the engineer understand the cause of the issues and focus on the correct part of the system.

3.9 Commissioning Report Integration

The final output of the method is a single commissioning report that combines all results produced during the analysis. All data is integrated into one JSON-based structure that contains the full analysis result for the system. The data is visualised through an HTML-based report interface.

The report is constructed from the analysed dataset, where each tag already contains its evidence data, rule-based findings, journal entries, severity information and explanations. In addition, entity-level results and peer summaries are included in the same structure.

The integration process does not introduce new analysis. Its purpose is to organise the existing data into a consistent report format. All information is preserved, and no

findings are removed or filtered during this stage. This ensures that the report reflects the complete result of the analysis.

The structure of the report follows the system hierarchy. Tags are grouped under their corresponding entities, and both tag- and entity-level information are available at the same time. This allows the engineer to navigate the results in the same way as the system is organised.

The use of a single report is a design decision. Instead of producing separate outputs for different analysis, all results are integrated into one report. This avoids fragmentation of information and ensures that the engineer can work from a single source. The outcome is a commissioning report that contains all the relevant information required for system-level troubleshooting.

4 Case Study and Results

This chapter presents the application of the proposed analysis method in an industrial automation system and evaluates its behaviour in practical use. The results are analysed in two phases: pre-commissioning and commissioning.

In the pre-commissioning phase, the analysis is limited to configuration and structural data, and the focus is on identifying inconsistencies between system definitions. During commissioning, runtime data becomes available, allowing the analysis to incorporate observed tag-level and entity-level behaviour.

4.1 Case Environment and Evaluation Approach

The case study was conducted in a real industrial automation environment consisting of one SCADA system, several plants, and multiple PLCs. Due to the timing of the industrial project and this work, a controlled before-and-after comparison of the same plant was not available. Therefore, the case study uses two reports from the same SCADA environment as an alternative evaluation setup: one from pre-commissioning and one from commissioning.

The two plants are not identical, but they are not separate systems either. They use the same SCADA environment and, where possible, similar PLC structures. In practice, this means that the same DDT structures have been reused as much as possible. At the same time, the plants differ in size, signal structure, and complexity. This is important for the case study, because the method should not depend on one specific plant structure.

The first report represents the pre-commissioning phase. This report was generated from the larger and more complex plant. At this stage, live commissioning had not yet

started and therefore a live commissioning report from this plant was not available at the time of writing. The report is still relevant because it shows how the method can be applied before live commissioning, when the focus is on configuration data, structure, and completeness of available system information.

The second report represents the commissioning phase. This report was generated from a smaller plant where runtime data was available. This makes it possible to examine how the same method works when signal values can be included in the analysis.

The reports are therefore used to examine the method in two commissioning situations within the same SCADA environment. The pre-commissioning report tests the method against a larger and more complex automation structure. The commissioning report shows how the method works when runtime data is available.

The evaluation focuses on the reports generated. The purpose is to examine whether the reports support commissioning work by presenting collected evidence, detected findings, explanations, and entity-level context in a form that can be inspected by an engineer. The following sections first discuss the pre-commissioning report, then the commissioning report, and finally compare the role of the method in these two situations.

4.2 Pre-Commissioning Analysis

The pre-commissioning report was generated from the larger plant where live commissioning had not yet started. The plant had not been operated as a live plant, and real runtime history was not available. The available execution context was based on engineering data and PLC simulation.

In this phase, the analysis could not be used to inspect process behaviour. Instead, the report was used as a programming-phase check. The purpose was to compare PLC-side and OPC-visible signal structures against the SCADA and SQL-based runtime structures, and to show which expected signals were not yet present in the supervisory system.

4.2.1 Role of analysis in pre-commissioning

In the pre-commissioning case, the analysis was used before real commissioning had started. The plant was still under PLC and SCADA engineering, so the report was not used to judge process operation or plant readiness. Its role was to check whether the engineering chain was built consistently.

The report contained 137 entities and 877 tags. Of these, 57 entities and 287 tags had high-severity findings. In this case, these findings should not be read as 287 faults in an operating plant. They mainly show that the SCADA-side implementation was still incomplete at the time of the report.

The main question in this phase was whether the expected PLC-side and OPC-visible signals were also present in the SQL-based runtime structures. If a signal was visible through OPC but missing from the runtime-side data, the report marked it as a mismatch. This gives the SCADA programmer a direct list of signals that required checking.

The role of the analysis was therefore practical. It reduces manual comparison between PLC engineering data, OPC browse results, and runtime database content. Instead of checking the signal chain one tag at a time, the report collected the missing or mismatching items into one engineering report.

4.2.2 Detection of configuration mismatches

The main configuration mismatch in the pre-commissioning report was related to signals that were visible through OPC but were not found from the SQL-based runtime structures. These findings were reported as *OPC_SQL_TAG_NOT_FOUND*.

This means that the signal had reached the OPC, but the corresponding runtime-side representation was not found from the supervisory system data. In this phase, the plant was still under engineering, so the finding was interpreted as an open SCADA/runtime implementation item rather than a process fault.

The report contained 287 findings of this type. The result shows that a large part of the PLC and OPC signal structure had already been created, but the same structure had not yet been fully implemented on the supervisory side. This is expected in an unfinished programming phase, but it is still useful because the report points to the exact tags that require checking.

For the SCADA programmer, the finding narrows the work to concrete configuration items. The missing runtime-side representation may be caused by an object that has not yet been deployed, a tag that has not been generated, an incomplete import, or a naming difference between OPC and runtime data.

The same principle also applied to structured signals. When the PLC engineering data contained the expected structure, the report could show that the PLC-side definition existed even if the corresponding supervisory-side representation was still missing. In this case, the finding did not mean that the PLC-side structure was wrong. It meant that the SCADA/runtime side had not yet been completed to the same level.

4.2.3 Behaviour under incomplete or missing data

The pre-commissioning report was generated from incomplete data. This was not an exception in this case, but part of the test setup. The plant had not been live, and the supervisory-side implementation was still in progress.

History analysis was therefore not applicable. There was no real process to analyse, so missing history samples were expected. They were not treated as history collection faults, stuck signals, or evidence of process behaviour.

Missing runtime-side data was handled in the same conservative way. If a tag was not found from the SQL-based runtime structures, the report did not try to infer why it was missing. At this stage, the reason could simply be that the SCADA configuration had not yet been created.

4.2.4 Limitations of analysis in pre-commissioning phase

Pre-commissioning analysis prepares the system for live commissioning. It can reveal missing tags, missing runtime structures, and incomplete mappings before the real signal tests begin.

This matters especially in this case, because the plant was still under engineering. The report was useful for checking the PLC-OPC-SCADA chain, but it was not a finished pre-commissioning package. A signal may look correct in the engineering data and still require later testing with the real field device, wiring, and process state.

No live history was available, so behaviour-based analysis was outside the scope of this report. The peer summary was generated, but it had little practical value. Peer

comparison becomes practically useful during commissioning, when real runtime data is available.

The result is therefore mainly a preparation aid. The report helps to remove configuration gaps before commissioning, so that live testing can focus on the actual plant instead of missing supervisory-side structures.

4.3 Commissioning Analysis

The commissioning report was generated from the smaller plant where runtime data was available. In this phase, the analysis could therefore include current OPC values together with configuration and SQL-based information.

This section examines how the report supported commissioning work when live runtime data was available. The following subsections discuss the role of the analysis during commissioning, signal-level and entity-level observations, peer comparison, and combined interpretation for troubleshooting.

4.3.1 Role of analysis during commissioning

During commissioning, the automation system is tested with live communication and runtime data. The analysis can therefore use current OPC values, not only configuration data. This makes it possible to check whether signals are visible through OPC, and whether their live values can be read.

The analysis also uses SQL-based runtime and history information. This is important because a signal being visible in OPC does not yet mean that the whole system chain is

working. For commissioning and later troubleshooting, the signal must also exist in the database structures and, when required, produce history values. If history is missing, the signal may still be visible live, but it cannot be verified later from stored data.

Commissioning is also a phase where the system can still change. PLC logic, signal definitions, and data structures may be corrected when faults or missing functions are found. After such changes, it is not enough to check only the PLC side. The change must also be visible in OPC, SCADA, SQL-based data, and history collection if the signal is expected to be stored in history.

For this reason, the analysis is used to support checking the signal chain after changes. It helps to see whether a signal exists in the expected structures, whether it can be read live, whether it is represented in SQL-based data, and whether history collection is working. The report does not replace manual commissioning tests, but it gives a structured way to find where the chain may be incomplete.

In this phase, the main value of the analysis is that it brings these checks into one engineering report. The engineer can inspect live values, database information, history observations, findings, and explanations without checking each system layer separately.

This phase also includes peer summaries. During commissioning, these summaries are more useful than in pre-commissioning because the comparison can use live runtime values. The peer summary shows whether one entity differs from comparable entities in the same environment. For example, it can highlight differences in operating bands or other entity-level values that may require engineering review. It does not replace the rule-based findings, but it gives the engineer an additional way to inspect configuration-level and entity-level differences together with the rest of the report.

4.3.2 Signal-level and entity-level observations

The commissioning report contained 129 entities and 715 tags. Most of the analysed system was classified as informational. Only two entities and two tags had high-severity findings, and three entities and three tags had medium-severity findings. This shows that the analysed signal chain was mostly working from the field level to the supervisory level. PLC, OPC, SCADA, and SQL-based data could be read and combined in the report, and there were no signs of a system-wide communication problem.

One clear signal-level finding was related to *OL1_PM5560.Current_G*. The signal belonged to the *OL1_PM5560* entity, which contained 36 tags in the report. The other tags of the same device were available, but the ground current signal did not provide a usable value. This made the finding more specific. The issue did not indicate that the whole device or communication path was missing. Instead, it indicated that the ground current value itself was not available from this device.

This observation was useful from a commissioning point of view. Because the other signals of the same device were working, the next action did not need to focus on the whole device communication. The finding pointed to one specific signal. If the device does not provide this value in the actual system, the signal can be removed or corrected in the PLC and SCADA configuration. In this case, the report helped separate a single unavailable measurement from a wider communication problem.

The report also produced observations related to history collection. Some Word bit signals did not have history values, although other signals in the same entity had history. These findings were related to *OL1_SC105.HalytysSana*, *OL1_UIQ004.StatusWord*, and *OL1_UIQ211.StatusWord*. This was useful because the report did more than state that history was missing. It showed that history was missing for a specific part of an entity where other signals were already being stored in history. *OL1_UIQ004.StatusWord* is shown in Figure 3.

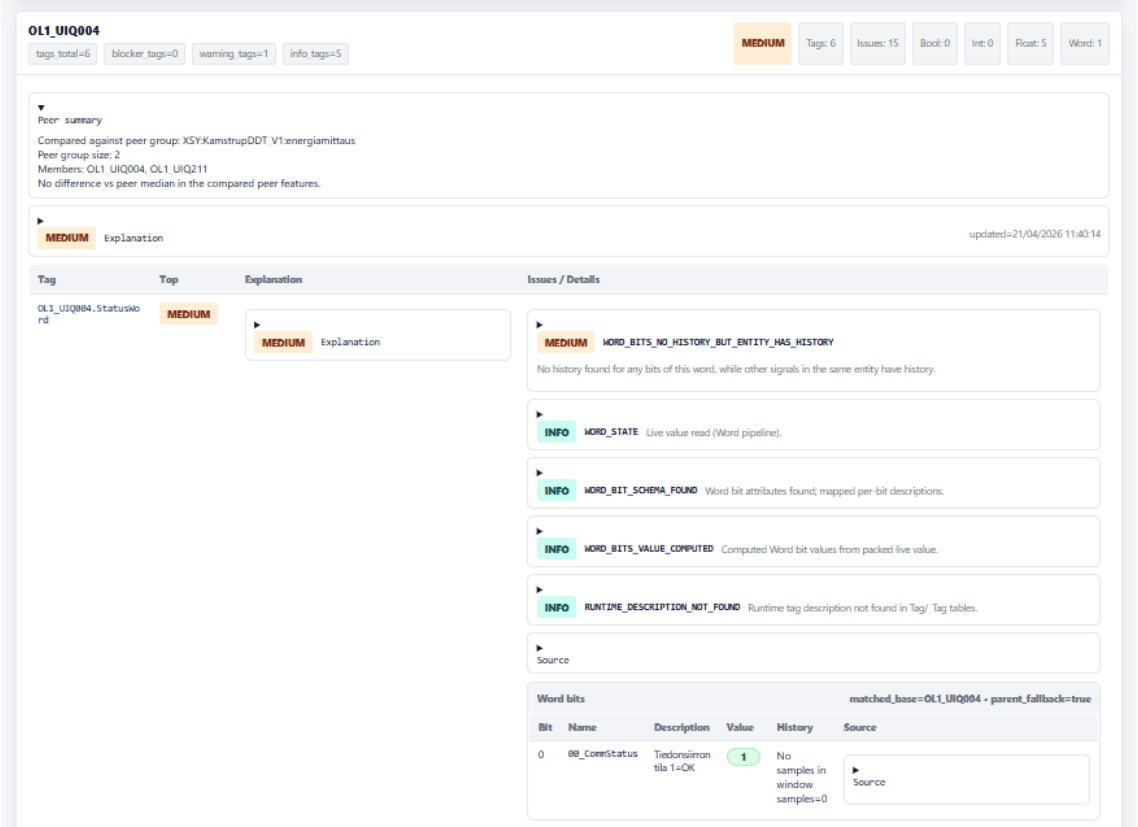

Figure 3. Example of a Word bit history finding in the commissioning report.

This type of finding is important during commissioning because live visibility alone is not enough. A signal may be readable through OPC, but it must also be stored in history if it is needed for later verification or troubleshooting. The Word bit findings therefore pointed to history configuration checks rather than to PLC logic or live communication.

The report also showed that SQL access itself was working. The SQL-related evidence did not return query errors. This means that missing SQL or history information should not first be interpreted as a failed SQL query. It is more likely that the missing information is related to configuration, naming, history collection, or system-level implementation. This helps direct troubleshooting to the correct part of the system.

At entity level, the report helped to interpret whether a finding was isolated or part of a wider pattern. The *OL1_PM5560.Current_G* case showed an isolated signal issue inside an otherwise readable entity. The Word bit history findings showed a different type of

issue, where a specific category of signals inside an entity required history checks. In both cases, the entity-level view made the result more useful than a flat tag list, because it showed where the finding belonged and what kind of follow-up action was reasonable.

4.3.3 Use of peer comparison in runtime context

The peer comparison produced two useful result types in the commissioning report. It showed cases where comparable entities were consistent with each other, and it highlighted configuration differences in measurement entities.

One consistent peer group was formed by *OL1_UIQ004* and *OL1_UIQ211*. These were comparable energy measurement entities, and the report did not show deviations from the peer median in the compared features. This indicated that the two entities were similar in the selected comparison features.

The clearest peer findings were related to configured operating bands. In this context, operating band means the difference between the configured high and low limits of a measurement. A different operating band does not automatically mean that the configuration is incorrect, because the measurements may have different process roles. However, it shows which signals should be checked manually during commissioning.

For pressure measurements, *OL1_PT209* and *OL1_PI301* were compared in the same peer group. The peer median for the operating band was 2.88. *OL1_PT209* had a lower operating band of 2.25, while *OL1_PI301* had a higher operating band of 3.5. This showed that the two pressure measurements were not configured with identical ranges. The report therefore pointed to a concrete configuration difference that could be checked against the intended measurement ranges.

For temperature measurements, one peer group included *OL1_TI101*, *OL1_TI201*, and *OL1_TT205*. The peer median for the operating band was 52.5. *OL1_TI201* had a much lower operating band of 12.5. This means that its configured high and low limits were much closer to each other than in the comparable temperature measurements. This may be correct if the signal has a different purpose, but it is a clear candidate for engineering review. *OL1_TI201* is presented in Figure 4.

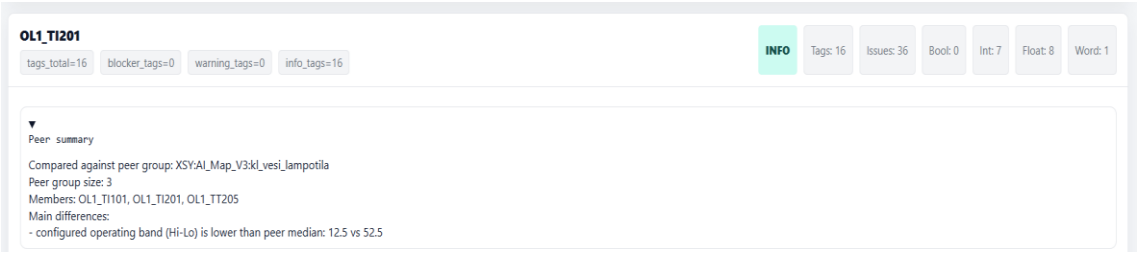

Figure 4. Example of peer-level comparison in the commissioning report.

Another temperature-related example was the peer group containing *OL1_TIC002* and *OL1_TIC107*. The peer median for the operating band was 46.75. *OL1_TIC002* had a lower operating band of 35, while *OL1_TIC107* had a higher operating band of 58.5. This again showed a configuration difference between comparable measurements.

The peer comparison therefore produced both confirming and deviating observations. In this report, the clearest deviations were related to configured operating bands in pressure and temperature measurements.

4.3.4 Combined interpretation for troubleshooting

The commissioning report allowed findings to be interpreted together instead of as separate tag-level observations. This was important because the same issue can look different depending on whether it is inspected through OPC, SQL-based data, history collection, or entity context.

The *OL1_PM5560.Current_G* finding was one example of this. At signal level, a usable ground current value was not available. At entity level, the same device contained 36 tags, and the other tags were available. This changed the interpretation of the finding. The issue did not indicate a device-level or communication-level problem. Instead, it pointed to one unavailable measurement. The next check could be directed to the device data, PLC configuration, and SCADA configuration for that specific signal.

The Word bit history findings required a different troubleshooting path. In these cases, the issue was not live communication. The relevant observation was that history values were missing for specific Word bit signals, although other signals in the same entities had history. This directed the check towards history configuration rather than PLC logic or OPC communication.

The peer comparison added configuration-related observations to the same report. The pressure and temperature findings showed differences in configured operating bands between comparable measurements. These differences were not direct fault conclusions, but they marked values that could be checked against the intended measurement ranges.

The combined interpretation helped separate different types of follow-up actions. From one report, the engineer could inspect the signal itself, the related entity, SQL-based information, history observations, and peer comparison results. This made it easier to see whether the next check should focus on one signal, history collection, configuration limits, or a wider part of the system chain. As a result, the report was more useful for troubleshooting than a separate list of individual findings.

4.4 Comparison Between Pre-Commissioning and Commissioning

The two reports were generated from different project phases, and this changed how the results were interpreted. In pre-commissioning, the report was used before live signal testing. The focus was on whether the PLC-side and OPC-visible signals had corresponding SCADA/runtime structures. During commissioning, live runtime data was available, so the same analysis and report format could also be used to inspect current signal values and history collection.

The clearest difference was the meaning of data. In pre-commissioning, missing runtime-side structures and missing history were expected because the programming phase was still ongoing. During commissioning, missing data was more specific. If a live signal was expected to be stored in history but no history was found, the finding pointed to a history configuration check.

The severity counts also had to be read differently. The pre-commissioning report contained more high-severity findings due to unfinished programming phase. In the commissioning report, there were fewer high- and medium-severity findings, and these could be interpreted more directly because the system chain was already mostly available.

Peer comparison has limited use before commissioning due to lack of real OPC values. During commissioning, peer comparison was more useful, and it was possible to inspect differences such as operating band deviations between similar measurements.

The comparison shows that the role of the analysis depended on the project phase. Before commissioning, it was mainly a configuration and completeness check. During commissioning, it became a troubleshooting aid that combined live values, history observations, entity context, and peer comparison. Table 3 summarises the results pre-commissioning versus commissioning.

Table 3. Pre-commissioning and commissioning result comparison.

Metric	Pre-commissioning	Commissioning
Entities analysed	137	129
Tags analysed	877	715
High severity entities	57	2
High severity tags	287	2
Medium severity entities	0	3
Medium severity tags	0	3
Runtime availability	Limited	Available
Peer comparison usefulness	Limited	More useful

5 Discussion

This chapter discusses the case study in relation to the research questions. The focus is on how the hybrid analysis approach supported commissioning, how simulated data can be used during development, and how explainability affected the interpretation of the results.

The discussion also considers the limitations observed in the case study, especially the difference between configuration-based analysis and live commissioning data. The following sections discuss each research question separately.

5.1 Research Question 1 – Hybrid Approach Effectiveness

The first research question asked how a hybrid and explainable analysis approach can support fault detection during commissioning. Based on the case study, the answer is that the hybrid structure was useful because the two parts did different work. The rule-based part handled the main signal-chain checks, while the machine learning part added peer-level context when data was complete enough for comparison.

The case study also showed that the hybrid approach relied mainly on rule-based analysis. This was expected, because the most relevant commissioning findings were concrete integration and configuration problems rather than patterns that needed to be learned from data. Missing mappings, missing runtime structures, unavailable values, and incomplete history collection could be checked directly against the available engineering evidence.

The machine learning component was useful in a narrower situation. It helped when entities were implemented far enough to be compared with each other. In that case,

peer comparison could show configuration differences that were not visible from one tag alone. In the unfinished pre-commissioning case, this condition was not met, so the peer summary did not have much practical value.

The effectiveness of the hybrid approach came from this separation. Rule-based checks produced the main findings. Peer comparison added a second view when the data supported it. The report did not decide the final cause of a fault, but it helped to narrow where the next engineering check should be done.

5.2 Research Question 2 – Simulated Data

The second research question asked to what extent simulated entity-level training cases can be used to support the training and model structural validation of the machine learning component during the development phase. Based on the case study, the answer is that simulated data had a limited but useful role. It supported the training and structural validation of the decision tree used in the peer-level analysis, but it could not prove the practical reliability of the whole commissioning support method.

In this context, simulated cases do not refer to physical process faults or equipment failures. They refer to artificial entity-level training cases used when creating the synthetic peer-relative dataset.

In this work, simulated data was used to create a synthetic peer-relative dataset for training the decision tree used in the peer-level analysis. The dataset consisted of artificial peer groups where most entities represented typical peer behaviour, while some entities were made atypical by changing selected entity-level features.

The simulated features described simplified entity-level conditions. These included, for example, finding counts, missing live values, pipeline and semantic role counts, OPC and

type-related issue counts, runtime and history availability indicators, Word bit indicators, and configuration-related values such as scale spans, operating bands, and delay sums. Peer-relative features were created by comparing selected entity values against the peer group median.

This means that simulated data was mainly useful for training the decision tree to distinguish typical and atypical peer behaviour in a controlled dataset. It supported the technical development of the peer classification mechanism, but it did not represent the full commissioning environment. It did not model the complete PLC, OPC, SCADA, SQL, and history chain in the same way as real project data.

The actual commissioning analysis was based on real project and runtime data. Entity features, peer groups, peer medians, peer differences, findings, and history observations were formed from the real report data. The machine learning component was used only as supporting information. It did not override rule-based findings and did not create new fault-level conclusions.

For this reason, real reports were still needed to evaluate the method in commissioning conditions. Real commissioning data contains project-specific naming, incomplete configurations, missing values, history collection issues, and system-layer behaviour that cannot be fully reproduced with synthetic data.

In this study, the decision tree component was not evaluated as an independent fault classifier using metrics such as accuracy, recall, or a confusion matrix. A sufficiently large labelled ground truth dataset was not available for that type of evaluation. Instead, the evaluation focused on whether the peer-level analysis produced traceable and engineering-relevant observations in the real commissioning report. The output was reviewed through the decision tree structure, feature importance, and the peer summaries generated for the analysed entities.

5.3 Research Question 3 – Explainability

The third research question asked how a hybrid and explainable analysis approach supports commissioning engineers in interpreting signal-level and logical anomalies. Based on the case study, the answer is that the approach supported engineers by making the analysis traceable and reviewable. The report helped the engineer understand why a finding was raised, what part of the system it referred to, and what kind of follow-up check it pointed to.

In this study, explainability means that the result can be traced back to the signal, entity, evidence, and comparison that produced it. This was important because the report was not intended to be a black-box fault list. The engineer needed to see why a finding was raised and what part of the system it referred to.

The *OL1_PM5560.Current_G* case showed this at signal level. The report explained the high-severity finding with the text: “OPC value is not finite (NaN/Inf); cannot trust signal.” This explanation stated the immediate technical reason for the finding. When it was interpreted together with the entity context from Section 4.3.2, the result was not only that one tag had a problem. It was possible to see that the issue was limited to one unavailable measurement in an otherwise readable device.

The Word bit history findings showed a different kind of explanation. The report stated: “No history found for any bits of this word, while other signals in the same entity have history.” This explanation was useful because it described why the finding mattered in that specific entity. The report also showed that the Word bit structure had been found and that bit values could be computed from the packed live value, which separated live signal interpretation from history collection.

The peer summaries showed the same principle in the decision tree-based part of the analysis. For example, the report explained that *OL1_PT209* had a “configured operating band (Hi-Lo)” lower than the peer median, 2.25 compared with 2.88. In another peer result, *OL1_TI201* was marked with an unusual operating band, 12.50 compared with the peer value 52.50. These explanations were useful because they showed the compared feature, the direction of the difference, and the peer reference value. The result was therefore not only a label saying that an entity differed from its peers, but a comparison that the engineer could review against the intended configuration.

These examples show that explainability in this work was not limited to the decision tree model itself. It was also created by the way rule-based findings, entity context, evidence, and peer summaries were presented together in the report. This is important in commissioning, because the engineer must be able to decide whether a finding is an actual issue, an expected configuration difference, or a situation that only requires further checking. The value of the approach was therefore not automatic decision-making, but traceable support for engineering review.

6 Limitations and Future Considerations

This study has several limitations that should be considered when interpreting the results. The method was evaluated in one industrial environment, although the environment included several plants and PLCs. The case study therefore shows that the method can be applied in a real project context, but it does not prove that the same approach would work without changes in every automation environment.

The method is also limited by the data that is available during the project phase. If a signal is missing from PLC configuration, OPC, SQL-based data, or history collection, the report can show the missing information and place it in the related entity context. However, the report cannot always determine the original reason for the missing information. In these cases, the interpretation still requires project knowledge and engineering judgment.

Another practical limitation was the dependency on project-specific configuration rules. The PLC structures, signal naming conventions, and implementation practices were not fully uniform, and a significant amount of configuration was needed to classify and interpret the data correctly. This included rules for pipelines, Word signals, semantic roles, and other project-specific structures. The method was still able to handle the data, but the amount of required configuration showed that the quality and consistency of engineering data strongly affect how easily the analysis can be applied.

The method was intentionally designed as a commissioning support tool rather than as an autonomous commissioning system. Fully automatic commissioning decisions would require a much more complete model of the plant, including process behaviour, control logic, operating states, and intended functional relationships between signals. In practical commissioning work, the information is often project-specific and not available in a form that could be used reliably for automatic decision-making. For this reason, the

method focuses on collecting evidence, identifying inconsistencies, and explaining findings in a form that supports engineering judgement.

One future improvement would be to make the project phase explicit in the software configuration. The current implementation can be used in both pre-commissioning and commissioning situations, but the analysis phase is not yet selected as a separate configuration option. This could be useful because the same observation may have a different meaning depending on the project phase. For example, missing history values may be expected in pre-commissioning if live operation has not yet started. During commissioning, the same observation may be more important if the signal is expected to be stored in history. A configurable pre-commissioning or commissioning mode could therefore affect severity classification and prioritisation without changing the basic analysis logic.

The peer-level machine learning layer is another area for future development. In this work, the peer comparison focused on entity-level features and differences between structurally similar entities. This was useful for identifying configuration differences such as operating band deviations. A possible extension would be to analyse logical relationships between related signals such as process value/setpoint relationships, output/feedback relationships, and command/status pairs. However, these relationships are not always located inside a single entity, and they may depend on project-specific naming, control structures, and engineering conventions.

The commissioning report showed that live signal visibility and stored data availability are separate issues. Some signals were visible through OPC and had stored history values, while others were visible live but did not have history values. Since the implementation separates missing history data from SQL query failures, the report could show that this was not an SQL access problem. This made the uneven history collection visible during commissioning and helped direct the follow-up towards history configuration.

7 Conclusion

This study examined how fault detection during commissioning of industrial automation systems can be supported with a hybrid and explainable analysis approach. The work started from a practical problem: during commissioning, many faults are not failures of physical equipment. They are missing mappings, incomplete signal chains, unavailable runtime values, missing history collection, or inconsistencies between PLC, OPC, SCADA, and database structures. These problems are difficult to handle with methods that assume stable operation, completed process history, or an accurate process model.

The work addressed the research gap identified in the introduction. It showed that real project configuration and runtime data can be used for commissioning-focused analysis, even when the system is still incomplete.

The developed method worked best as an engineering support method. Rule-based checks formed the main part of the analysis, because the relevant findings could be checked directly from available evidence. The machine learning component had a smaller role. It supported peer-level comparison when enough comparable data was available, but it did not replace the rule-based findings or make final fault decisions.

The case study showed that the same analysis approach can be used before and during live commissioning, but the findings must be interpreted according to the project phase. Before live commissioning, the report mainly helped to find missing supervisory-side structures and incomplete mappings. During commissioning, live values and history observations made the findings more useful for troubleshooting. A missing item therefore did not always mean the same thing. In an unfinished system it could be open implementation work, while in live commissioning it could point to a configuration item that should be checked.

The report made the findings usable by showing them together with evidence, entity context, explanations, and peer information when available. This helped the engineer see whether the next check should focus on one signal, history collection, a missing runtime structure, or a configuration difference between comparable entities.

With the defined scope, the objective of this work was achieved. The method supported commissioning fault detection by reducing scattered manual checking across system layers and by showing where the next engineering check should be made. It did not remove the need for manual commissioning or engineering judgement, but it made the available evidence easier to inspect.

The results also show that the method can be developed further. Future work should focus on making the project phase explicit in the tool configuration, extending peer-level analysis towards logical relationships, and adapting the method to environments with different data sources, communication protocols, or database structures. These extensions would not change the purpose of the method. The goal would remain to support commissioning engineers with a unified engineering report.

References

Arp, D., Quiring, E., Pendlebury, F., Warnecke, A., Pierazzi, F., Wressnegger, C., Cavallaro, L., & Rieck, K. (2020). *Dos and don'ts of machine learning in computer security* (arXiv:2010.09470). arXiv. Retrieved 15 December 2025, from https://arxiv.org/abs/2010.09470

Blázquez-García, A., Conde, A., Mori, U., & Lozano, J. A. (2021). A review on outlier/anomaly detection in time series data. *ACM Computing Surveys, 54*(3), Article 56. Retrieved 15 December 2025, from https://doi.org/10.1145/3444690

Braga-Neto, U. (2024). *Fundamentals of pattern recognition and machine learning* (2nd ed.). Springer. Retrieved 15 December 2025, from https://doi.org/10.1007/978-3-031-60950-3

Brusa, E., Cibrario, L., Delprete, C., & Di Maggio, L. G. (2023). Explainable AI for machine fault diagnosis: Understanding features' contribution in machine learning models for industrial condition monitoring. *Applied Sciences, 13*(4), Article 2038. Retrieved 23 February 2026, from https://doi.org/10.3390/app13042038

Chamola, V., Hassija, V., Sulthana, A. R., Ghosh, D., Dhingra, D., & Sikdar, B. K. (2023). A review of trustworthy and explainable artificial intelligence (XAI). *IEEE Access, 11*, 78994–79015. Retrieved 15 December 2025, from https://doi.org/10.1109/ACCESS.2023.3294569

Hamrouni, I., Lahdhiri, H., Ben Abdellafou, K., Aljuhani, A., & Taouali, O. (2023). Anomaly detection for process monitoring based on machine learning technique. *Neural Computing and Applications, 35*, 7889–7906. Retrieved 15 December 2025, from https://doi.org/10.1007/s00521-022-07901-2

Isermann, R. (1997). Supervision, fault-detection and fault-diagnosis methods—An introduction. *Control Engineering Practice, 5*(5), 639–652. Retrieved 23 February 2026, from https://doi.org/10.1016/S0967-0661(97)00046-4

Isermann, R. (2005). Model-based fault-detection and diagnosis—Status and applications. *Annual Reviews in Control, 29*(1), 71–85. Retrieved 23 February 2026, from https://doi.org/10.1016/j.arcontrol.2004.12.002

ISO/IEC/IEEE. (2022). *Software and systems engineering—Software testing—Part 1: General concepts* (ISO/IEC/IEEE 29119-1:2022). International Organization for Standardization. Retrieved 23 February 2026, from https://www.iso.org/standard/81291.html

Lu, J., Liu, A., Dong, F., Gu, F., Gama, J., & Zhang, G. (2019). Learning under concept drift: A review. *IEEE Transactions on Knowledge and Data Engineering, 31*(12), 2346–2363. Retrieved 15 December 2025, from https://doi.org/10.1109/TKDE.2018.2876857

Lundberg, S. M., Erion, G., Chen, H., DeGrave, A., Prutkin, J. M., Nair, B., Katz, R., Himmelfarb, J., Bansal, N., & Lee, S.-I. (2020). From local explanations to global understanding with explainable AI for trees. *Nature Machine Intelligence, 2*, 56–67. Retrieved 15 December 2025, from https://doi.org/10.1038/s42256-019-0138-9

Moreno-Torres, J. G., Raeder, T., Alaiz-Rodríguez, R., Chawla, N. V., & Herrera, F. (2012). A unifying view on dataset shift in classification. *Pattern Recognition, 45*(1), 521–530. Retrieved 23 February 2026, from https://doi.org/10.1016/j.patcog.2011.06.019

PLCopen. (2016, October). *IEC 61131-3: A standard programming resource*. Retrieved 23 February 2026, from https://www.plcopen.org/download_file/force/bcddc27b-53f6-44ea-b0e3-e3d49c1564dc/342/

Sahu, A. R., Palei, S. K., & Mishra, A. (2024). Data-driven fault diagnosis approaches for industrial equipment: A review. *Expert Systems, 41*(2), e13360. Retrieved 15 December 2025, from https://doi.org/10.1111/exsy.13360

Schamp, M., Van De Ginste, L., Hoedt, S., Claeys, A., Aghezzaf, E.-H., & Cottyn, J. (2020). Virtual commissioning of industrial control systems – A 3D digital model approach. *Procedia Manufacturing, 51*, 1343–1350. Retrieved 24 February 2026, from https://doi.org/10.1016/j.promfg.2020.01.229

Sculley, D., Holt, G., Golovin, D., Davydov, E., Phillips, T., Ebner, D., Chaudhary, V., Young, M., Crespo, J.-F., & Dennison, D. (2015). Hidden technical debt in machine learning systems. In C. Cortes, N. D. Lawrence, D. D. Lee, M. Sugiyama, & R. Garnett (Eds.), *Advances in Neural Information Processing Systems* (Vol. 28, pp. 2503–

2511). Curran Associates, Inc. Retrieved 23 February 2026, from https://papers.nips.cc/paper_files/paper/2015/file/86df7dcfd896fcaf2674f757a2463eba-Paper.pdf

Shen, Y., Ding, S. X., Xie, X., & Luo, H. (2014). A review on basic data-driven approaches for industrial process monitoring. *IEEE Transactions on Industrial Electronics, 61*(11), 6418–6428. Retrieved 23 February 2026, from https://doi.org/10.1109/TIE.2014.2301773

Velesaca, H. O., Holgado-Terriza, J. A., & Gutierrez-Guerrero, J. M. (2025). Industrial process automation through machine learning and OPC-UA: A systematic literature review. *Electronics, 14*(18), Article 3749. Retrieved 15 December 2025, from https://doi.org/10.3390/electronics14183749

Venkatasubramanian, V., Rengaswamy, R., Yin, K., & Kavuri, S. N. (2003a). A review of process fault detection and diagnosis: Part I—Quantitative model-based methods. *Computers & Chemical Engineering, 27*(3), 293–311. Retrieved 23 February 2026, from https://doi.org/10.1016/S0098-1354(02)00160-6

Venkatasubramanian, V., Rengaswamy, R., Yin, K., & Kavuri, S. N. (2003b). A review of process fault detection and diagnosis: Part II—Qualitative models and search strategies. *Computers & Chemical Engineering, 27*(3), 313–326. Retrieved 23 February 2026, from https://doi.org/10.1016/S0098-1354(02)00161-8

Venkatasubramanian, V., Rengaswamy, R., Yin, K., & Kavuri, S. N. (2003c). A review of process fault detection and diagnosis: Part III—Process history based methods. *Computers & Chemical Engineering, 27*(3), 327–346. Retrieved 23 February 2026, from https://doi.org/10.1016/S0098-1354(02)00162-X