Int. J. Appl. Math. Comput. Sci., 2026, Vol. 36, No. 2, 329–345
DOI: 10.61822/amcs-2026-0022

AN EDGE–NATIVE INDUSTRIAL VERIFICATION OF REQUEST SYSTEM FOR SECURE AND PLAUSIBLE CONTROL RECONFIGURATION IN MODULAR PLANTS

YUANCHEN ZHAO[a,*], HEIKO SCHOON[b], TORBEN MINY[a], PATRICK FELKE[b], TOBIAS KLEINERT[a]

[a] Chair of Information and Automation Systems for Process and Material Technology
RWTH Aachen University
Turmstrasse 46, 52064 Aachen, Germany
e-mail: {y.zhao,t.miny,kleinert}@iat.rwth-aachen.de

[b] Department of Electrical Engineering and Computer Science
University of Applied Sciences Emden/Leer
Constantiaplatz 4, 26723 Emden, Germany
e-mail: {heiko.schoon,patrick.felke}@hs-emden-leer.de

With the rise of Industry 4.0, the convergence of information technology (IT) and operational technology (OT) has become a critical research area. While IT solutions enhance the connectivity and intelligence of OT systems, they also introduce significant cybersecurity risks to industrial environments. Simultaneously, modular design of plants has emerged as a scalable and reconfigurable paradigm for industrial production, enabling faster deployment and greater adaptability to changing process requirements. However, integrating IT functionalities into such modular systems for remote process monitoring and control reconfiguration in response to dynamic production recipes requires addressing trust, control, and system integrity. In this context, this paper proposes an edge-native industrial verification system to securely interface IT technologies with modular plant operations. The system automatically verifies and authorizes control reconfiguration requests, ensuring only legitimate, contextually plausible actions are executed within modular plants. The system is based on the Namur Open Architecture (NOA) but extends its concept by incorporating features specifically designed for modular plants, such as endpoint mapping based on the Module Type Package (MTP) and a data aggregation system to enhance scalability and interoperability. By embedding security and contextual verification at the edge, the system supports a trusted, adaptive cyber-physical infrastructure for robust, future-ready modular manufacturing systems.

Keywords: verification of request, secure IT/OT convergence, control reconfiguration in modular plants, industrial automation system.

1. Introduction

The advent of Industry 4.0 has significantly reshaped the landscape of industrial automation by driving the convergence of information technology (IT) and operational technology (OT). This convergence aims to improve the productive efficiency of the entire manufacturing value chain by leveraging advanced IT technologies to empower OT systems, specifically in areas such as data-driven decision-making and analytics, process optimization, cloud-based control, and remote monitoring (Boyes *et al.*, 2018). However, this integration also introduces vulnerabilities that may expose OT systems to cyber threats, which can have serious consequences for both operational safety and system integrity. Cyber attacks have the potential to alter implemented process monitoring and control functionality, leading to production equipment downtime, operator injury, or environmental damage. Compromised security caused by these attacks can allow malicious manipulation of safety measures and trigger shutdowns of machines or production lines, leading to service denial (Hollerer *et al.*, 2021). The convergence of OT with IT systems has expanded the attack surface of automation systems and industrial plants, introducing new vulnerabilities. This cybersecurity challenge is evidenced by notable incidents such as the 2016 Industroyer malware

*Corresponding author

attack that targeted Ukrainian power grids by exploiting SCADA system vulnerabilities, or the Triton malware attack on a Saudi Arabian petrochemical plant (Ryu *et al*., 2024). These cases represent significant examples of cyber attacks specifically designed to compromise OT environments.

Concurrently, the shift towards modular plant designs has emerged as a critical trend in modern industrial production. Large-scale production units are economically unsuitable for innovative products with presumably short lifespans and unpredictable market dynamics (Baldea *et al*., 2017). In contrast, modular plants offer flexibility and scalability by enabling rapid deployment, reconfiguration, and adaptability to ever-changing production requirements (Zhao *et al*., 2024). This approach allows quick implementation of new processes and systems, facilitating the efficient management of resources and minimizing downtime. However, the integration of IT functionalities into these modular systems to enable remote process monitoring and control reconfiguration introduces a complex set of challenges related to trust, control, and system integrity. The need for a secure and reliable interface between IT technologies and OT systems becomes even more pronounced in modular plants, as real-time data interactions with the IT central server, remote control, and dynamic control reconfiguration of the plants in response to changes in production recipes occur frequently. Addressing these challenges requires a secure mechanism to verify and authorize control reconfiguration requests, ensuring that only legitimate and contextually plausible actions are executed within the modular plants.

To address this issue, the NAMUR Open Architecture (NOA) concept was proposed by the NAMUR association as a conceptual framework for integrating IT systems with process control ones, such as distributed control systems, in industrial environments (NAMUR, 2020). A major target of the NOA is to keep the integrity of core process control (CPC) when new IT technologies are introduced, making it especially well-suited for Industry 4.0 applications in brownfield plants. Overall, the core functionality of the NOA is realized through two key components: the data diode, which enables the unidirectional transmission of process data from the OT system to the IT one, and the verification of request (VoR) component, which transfers optimized data generated by IT-side applications back into the OT system in a controlled, unidirectional manner.

While there are already solutions for the data diode available (Honda *et al*., 2022; Azarmipour *et al.*, 2020b), there is a lack of solutions for VoR, particularly those applicable to modular plants. For this reason, this paper proposes an innovative edge-native industrial VoR system compliant with the NOA concept, designed to securely integrate IT technologies into modular plant operations, with a primary focus on ensuring system integrity and contextual plausibility of control reconfiguration in modular plants. The proposed VoR system not only serves as an interface for secure communication between IT and OT, but also provides the opportunity to leverage advanced IT technologies to optimize OT processes. Based on the optimization results, it enables secure and contextually plausible automatic control reconfiguration of modular plants, thereby alleviating the manual workload traditionally associated with control reconfiguration tasks and relieving plant operators from these responsibilities. Furthermore, as the entire control reconfiguration process is fully automated, both process optimization and control reconfiguration can be executed expeditiously, reducing downtime caused by control reconfiguration and minimizing the potential for human operational errors. In addition, the proposed VoR system not only fulfills the functional requirements defined by the NOA concept but also extends them by incorporating features tailored to the specific characteristics of modular plants, such as the addition of a data aggregation system (cf. Section 5.2), to enhance scalability and better accommodate modular plants. In summary, the proposed VoR system mitigates potential cybersecurity risks in the convergence of IT and OT, transforming traditional closed industrial OT systems into secure, open, and IT-interoperable OT ones, while enabling a trusted, adaptive, and robust cyber-physical infrastructure.

The rest of the paper is organized as follows. Section 2 presents the research goals and questions that guided the design and evaluation of the proposed VoR system. Section 3 reviews standards and technologies relevant to the system. Section 4 outlines the requirements defined by the SIoT-Gateway Project and the architecture of the proposed solution. Section 5 provides a detailed description of system implementation, including the technologies employed and the rationale behind their selection. Section 6 describes the study definition and evaluation methods used to systematically assess the functionality, correctness, and applicability of the proposed VoR system. Section 7 presents the integration of the system with the modular plant in the ModuLab laboratory and the control reconfiguration experiments conducted to validate its performance. Section 8 concludes the paper and outlines directions for future work. Additionally, to aid the reader's understanding, the XML-encoded rule sets used in the VoR system are provided in Appendices for reference.

2. Research goals and questions

The goal of this work is to realize a complete, NE 178-compliant (NAMUR, 2025) VoR system for the NOA. The proposed solution aims to ensure secure IT/OT request transfer and provide a systematic multi-level

verification of requests, covering the following:

- authentication and authorization (e.g., prevention of unauthenticated access and unauthorized configuration attempts),
- structural integrity (e.g., detection of malformed or empty requests),
- semantic mapping correctness (e.g., identification of undefined or unsupported request-to-parameter mappings),
- permission constraint validation (e.g., identification of unmet environmental or operational permission conditions such as required personnel presence), and
- parameter range verification (e.g., validation of configuration values against allowed operating limits).

Furthermore, the VoR system is designed to operate on a single hardware component, to support practical industrial operations such as modular plant reconfiguration, and to remain migratable across different industrial domains with minimal configuration effort.

Based on these goals, we formulate the following research questions (RQs).

RQ1: *Architectural completeness and feasibility.* To what extent can a single-hardware VoR architecture fully implement the authentication and authorization functions, structural-integrity checks, semantic-mapping validation, and permission and parameter-constraint verification required by NE 178?

RQ2: *Verification accuracy and functional behavior.* How accurately does the proposed VoR system detect the full range of NE 178 verification categories, including authentication or authorization violations, malformed requests, semantic-mapping issues, permission-constraint and parameter-range violations, while still accepting valid requests and providing meaningful, traceable feedback?

RQ3: *Cross-domain applicability and reusability.* Can the VoR system serve as a reusable and generalizable verification model that can be configured and adapted to different industrial domains with minimal configuration effort?

RQ4: *Migratability and adaptability in modular plants for reconfiguration.* Is the VoR system capable of supporting realistic workflows, such as module reconfiguration in a laboratory setup, in conjunction with the MTP standard, demonstrating its applicability to modular plant concepts?

These research questions define the scope of this study and guide the design and evaluation of the proposed VoR system. The next section provides the background and standards necessary to address them.

3. Background

This section reviews relevant standards and technologies and presents the state-of-the-art researches on VoR systems. In contrast to our previous work (Zhao *et al.*, 2025), which focused on introducing the overall architecture, this paper provides concrete implementation and validation of the proposed VoR system. It emphasizes the practical application of relevant technologies with enhanced security and flexibility considerations, extends the data integration capabilities based on the foundation laid by earlier work, and evaluates the system's performance in real-world scenarios.

3.1. NOA. The NOA concept was proposed to retain the conventional architecture of the automation pyramid within the context of CPC, ensuring alignment with established components of the OT domain. In addition to the CPC domain, the NOA introduces a monitoring and optimization (M+O) domain, which incorporates IT-based functionalities to support more adaptable and rapidly evolving applications. The M+O domain can be further divided into two categories: plant-specific M+O (psM+O) and central M+O. The psM+O component comprises monitoring and optimization functionalities that are exclusively tailored to the operational context of a single plant, including, for example, advanced process control. In contrast, the central M+O component integrates functions designed to support monitoring and optimization across multiple plants, e.g., at a site level. As this work focuses on a modular plant composed of multiple modular units, the M+O functionalities associated with individual units are classified as psM+O, while those concerning the plant as a whole are categorized as central M+O. Data diodes and VoR are two key components of the NOA that facilitate controlled bidirectional information exchange between the CPC and M+O domains.

3.1.1. NOA data diode. The data diode enables secure transmission of data without offering the possibility of sending data, information, commands or instructions back via the channel. In the current research, there are several approaches to implementing the data diode, including using OPC UA Publisher/Subscriber (Pub/Sub) (Honda *et al.*, 2022) and PikeOS queuing ports (Azarmipour *et al.*, 2020b). However, employing OPC UA Pub/Sub to implement a data diode presents challenges, including security vulnerabilities, configuration complexity, potential reliability issues, and difficulties in strictly enforcing one-way data flow. On the other hand, the approach using PikeOS's queuing port functionality to implement a data diode is highly dependent on the specific software architecture, which limits the solution's portability and general applicability across different

systems and environments.

3.1.2. NOA verification of request. VoR, proposed in NE 178, is a system-of-systems architecture that transports request from the M+O to the CPC domain. According to this standard, the functionality of VoR is divided into six clearly defined sequential steps: authentication and authorization, request verification, mapping, propagation, acceptance, and mapping verification. Compared to the data diode, which controls the flow of information out of the CPC domain, any information entering the domain must pass through the VoR functionality to ensure the legitimacy and plausibility of the request. Although the value of VoR has been recognized by both the industrial and academic communities in the field of process automation, research on its practical implementation remains limited and is mostly confined to proprietary commercial solutions, such as those offered by ABB and Emerson. Therefore, application-oriented and non-proprietary research in this area is crucial for continued advancement of the field.

Previous studies explored different approaches to implementing VoR. In the work of Azarmipour *et al.* (2020a), a digital twin was used for request validation, but this method lacks scalability due to the need for case-specific models. Pakala and Diedrich (2025) proposed a modeling of VoR based on the pro-active asset administration shell (AAS). However, the modeling suggested in this study remains incomplete, as its architectural design does not include the VoR propagation step. In addition, the work lacks details on how the virtual automation server performs the mapping, acceptance, and verification tasks. Furthermore, it does not clarify the rationale for assuming that the VoR server possesses complete knowledge of the plant structure, nor does it discuss the potential confidentiality risks or the implementation methods associated with this assumption. In the work of Grüner and Trosten (2023), a cloud-native architecture of NOA VoR using OPC UA Pub/Sub actions over MQTT (Message Queuing Telemetry Transport) was proposed. However, there, VoR functionalities are deployed on the Azure cloud infrastructure, exposing the VoR system to a broader IT environment and potentially introducing risks such as cyber attacks, privacy issues, and compliance concerns due to external dependencies and limited control. Additionally, the VoR steps are distributed across multiple devices, including Edge IPCs, VoR brokers, and CPC-DCS devices, which complicates maintenance and makes the solution difficult to adapt to other production environments. Furthermore, the solution relies exclusively on OPC UA communication, excluding OT devices that do not support it, a significant limitation given the widespread use of such devices in industrial environments.

3.2. OPC UA. The Open Platform Communications Unified Architecture, abbreviated as OPC UA, is a platform-independent standard designed for industrial network communication. Managed by the OPC Foundation, it provides a comprehensive framework for data and information exchange between components in industrial environments. At its core, the OPC UA supports two primary communication patterns: the Client/Server model and the Pub/Sub model (OPC Foundation, 2024).

The OPC UA provides structured mechanisms for user interaction through its method call service. Each method is described by a node in the OPC UA address space. This node contains the metadata identifying the method's arguments and describing its behaviour. These methods allow clients to invoke server-defined functions with input parameters and receive outputs, representing actions that can be performed on objects within the OPC UA address space.

Security is fundamental to the OPC UA's design, especially given its application across company hierarchies where communication disruption can lead to financial losses, safety hazards, or environmental damage (OPC Foundation, 2023).

The OPC UA is not secure by default: it requires a thorough configuration by system designers and consideration of domain-specific standards. The German Federal Office for Information Security (BSI) attested that specified security mechanisms provided by the OPC UA are sufficient to ensure secure communication according to current standards (FOIS, 2022). Compared to the Client/Server model, security in OPC UA Pub/Sub is more complicated and less fine-grained (Aro, 2021). While Client/Server security can enforce controls at the session and user level, Pub/Sub relies mainly on securing messages, limiting the ability to differentiate security policies for individual subscribers.

3.3. Message queue. A message queue is a mechanism that enables asynchronous communication between different software components or applications by transmitting messages through a shared buffer. This allows the sender to dispatch messages without waiting for an immediate response from the receiver, thereby improving system performance and enabling the decoupled operation of components (Maharjan *et al.*, 2023).

In order to identify appropriate mechanisms for message handling within the proposed VoR system, a set of functional requirements has been defined. These include support for priority-based message queuing, persistent yet lightweight data storage, and the capacity to handle concurrent access. In line with these criteria, this work conducts a comparative assessment of several representative message queuing solutions, namely, RabbitMQ, Redis, Apache Kafka, POSIX Message Queue

(POSIX MQ), and SQLite. The inclusion of SQLite in the message queue comparison is motivated by its use as a lightweight, file-based database system that can serve as a simple, persistent storage mechanism in embedded and local database scenarios, providing an alternative to more traditional message queue systems.

As summarized in Table 1, the chosen message queuing solutions exhibit distinct differences across several performance attributes, based on their technical documentation. In brief, RabbitMQ, Redis, and Apache Kafka exemplify feature-rich, widely adopted message-oriented middleware, while SQLite and POSIX MQ are considered lightweight alternatives, particularly suitable for deployment in embedded or resource-constrained environments.

3.4. Control reconfiguration in modular plants. The Module Type Package (MTP) and the associated standard originates from the VDI/VDE/NAMUR 2658 guideline. It is a standardized, vendor-independent description format that encapsulates capabilities, services, procedures, state machines and visualization elements of modular automation units, enabling their seamless integration into higher-level process control systems through a plug-and-produce approach. Through the MTP, increased system flexibility in the process industry is facilitated by enabling straightforward integration, reconfiguration, and scalability of modular components (Schmetz *et al.*, 2025).

However, when the need for control reconfiguration arises, it remains common practice to manually perform adaptations and extensions locally within the process control system. This requires operators to have sufficient background knowledge of the MTP to avoid errors during the reconfiguration process. Furthermore, the MTP is primarily focused on module-level modeling and integration, with limited support for system-wide behaviors, such as global optimization.

In the context of IT/OT convergence, the integration of secure edge gateway solutions with the MTP facilitates secure and efficient control reconfiguration in modular plants by offering a standardized, structured interface between the IT and OT systems. Through the MTP, OT data is structured in a standardized manner, enabling IT-generated control reconfiguration requests to be more effectively mapped to corresponding OT parameters. Moreover, the VoR system deployed on the secure Internet of things (SIoT) edge gateway ensures that any control reconfiguration request issued from the IT layer undergoes verification for legitimacy and plausibility before being applied to the OT system. Additionally, by integrating edge computing or artificial intelligence, the IT layer can perform real-time analysis of process data and update control parameters, and eventually structures, via the gateway, thereby enabling adaptive and autonomous process optimization and control reconfiguration. However, to the best of our knowledge, this concept has not yet been practically implemented or systematically explored, which constitutes one of the core focuses of this work.

4. Requirements and the proposed architecture

This section introduces relevant functional and non-functional requirements, which were defined in accordance with the SIoT-Gateway Project specification, based on the project's objectives and the security needs of the cloud-edge system. These requirements were identified through an analysis of the system architecture, relevant standards, and potential risks in data transmission and access control. The section also reviews the architecture proposed in our previous work (Zhao *et al.*, 2025).

4.1. Functional and non-functional requirements. The functional and non-functional requirements (FRs and NFRs) respectively, are essential for ensuring the system's effectiveness in both functionality and overall performance. The key requirements for the proposed VoR system, as defined by the SIoT-Gateway Project, are outlined as follows:

1. Functional requirements.

FR1: *Verification.* Direct communication from M+O to CPC is prohibited. Instead, requests from M+O must be verified by a VoR service before being sent to CPC.

FR2: *Unidirectionality.* From the CPC to M+O domain, unidirectional communication is mandatory to ensure that M+O cannot bypass the VoR system and directly issue change requests to CPC.

FR3: *Traceability.* The feedback generated by the VoR steps must be sent back to the request issuer in M+O to inform them of the current request processing result and logged for traceability.

FR4: *Preservation of existing security mechanisms.* The functionality of the VoR system should not interfere or downgrade existing security mechanisms in the CPC domain.

2. Non-functional requirements.

NFR1: *Confidentiality.* All sensitive data of the CPC components and the control reconfiguration requests must be encrypted during transmission and storage to prevent unauthorized information disclosure.

Table 1. Comparison of selected message queue mechanisms.

Feature	RabbitMQ	Redis	Apache Kafka	POSIX MQ	SQLite
Priority queue support	Yes	No	Limited	Yes	Yes
Message persistence	Yes	Yes	Yes	Partial	Yes
Resource efficiency	Medium-low	Medium	Low	Very high	High
Concurrency management	High	High	High	Medium	Low
Use case suitability	Microservices, large systems	Real-time, caching	Distributed, high volume	Embedded, lightweight	Embedded, local databases

NFR2: *Integrity.* Data must be protected against unauthorized modification, with mechanisms to detect tampering and ensure information accuracy.

NFR3: *Availability.* The system must remain operational and accessible to authorized users, with resilience against cyber attacks.

NFR4: *User authentication.* Applications and devices (clients) sending requests from M+O and their individual users (e.g., plant operator, administrator) connecting to the system must authenticate using certificates or secure credentials.

NFR5: *User authorization.* Authenticated and connected clients and their users must have defined access permissions to specific system resources and must be granted access based on their roles.

NFR6: *Flexibility.* Flexible deployment of applications within the M+O shall allow for configuration based on practical requirements.

NFR7: *Interoperability.* The system should seamlessly integrate with devices or components from different manufacturers without complex configuration and adjustments.

4.2. Proposed architecture. In our previous work (Zhao *et al.*, 2025), we proposed an architecture of a NOA-based SIoT edge gateway, which is designed to bridge the IT and OT domains while ensuring cybersecurity, enabling advanced IT technologies to empower process optimization in the OT domain.

Compared to other VoR system solutions, our approach offers the following advantages:

- Firstly, our solution integrates all of the VoR functionalities into a single edge device, enhancing portability, reducing hardware requirements, and simplifying subsequent maintenance and internal application updates.
- Secondly, our solution not only implements all the VoR functionalities proposed in NE 178 but also extends them by incorporating a data aggregation system. This allows frequent aggregation of data into a pool for other applications and serves as a data hub for interfaces to plant components and equipment. Furthermore, the integrated protocol conversion application enables the SIoT gateway to interface with devices that do not support the OPC UA communication protocol, thereby enhancing the overall interoperability and applicability of the proposed solution.
- Last but not least, the proposed solution features a dedicated deployment for control reconfiguration in modular plants. By utilizing MTP-defined parameters as the endpoint elements in VoR mapping, it not only improves adaptability to modular plant reconfiguration but also strengthens interoperability within the scope of this research and its intended application domain.

4.2.1. Partitioning. A detailed explanation of the functionalities of each partition can be found in our other work (Zhao *et al.*, 2025). In this paper, we briefly summarize the key aspects necessary to understand the presented implementation. Figure 1 illustrates the proposed architecture of the VoR system deployed on the SIoT edge gateway, which consists of three communication flows and four partitions. Depending on the type of message being transmitted, the information flow within the architecture can be classified into three categories: process data flow, which conveys real-time operational data; request flow, which carries control reconfiguration requests; and feedback flow, which returns request verification status from VoR steps. According to the sequential processing order of the request, the partitions are arranged as follows: the interface partition, the plant-specific M+O partition, the intermediate verification partition, and the CPC partition. The partitioning is determined based on the functional interdependencies of the VoR steps. If the execution of a subsequent step directly depends on the output of a preceding step, or if an update in one step necessitates corresponding modifications in the next, or if a step needs to share the same file system with the subsequent one, both are assigned to the same partition to ensure the continuity and integrity of the request verification process. When the logic of a VoR step needs to be

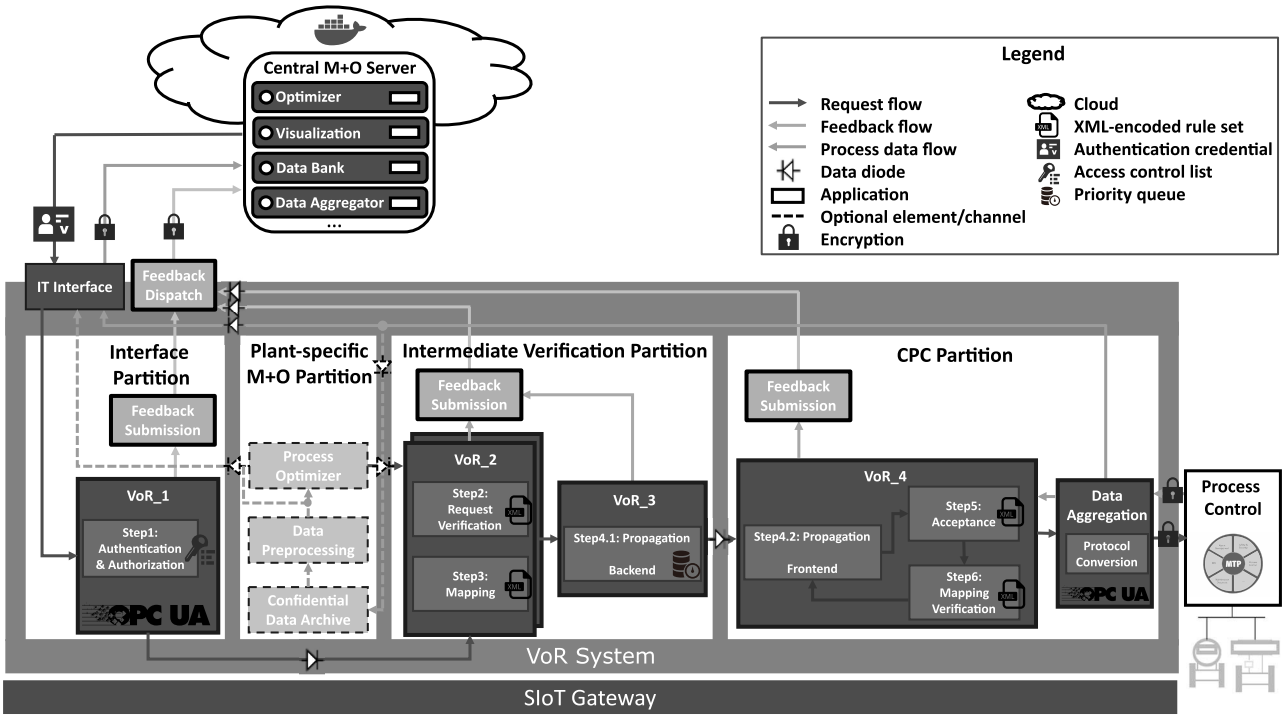

Fig. 1. Proposed architecture of the VoR system deployed on the SIoT edge gateway.

updated, all applications within the same partition should be considered first for corresponding updates to maintain system consistency and operational reliability.

It is worth mentioning that the plant-specific M+O partition is only deployed locally if sufficient resources are available on the edge gateway. Moreover, since only the VoR system manager has the authority to redeploy it, the process optimizer within the plant-specific M+O partition is by default regarded as a trusted entity and is therefore exempt from executing VoR Step 1.

4.2.2. VoR steps. The functionality of VoR is implemented within the proposed VoR system through the sequential execution of VoR Steps 1 to 6 (cf. Fig. 1). The steps and their execution sequence in this work are designed in accordance with the NE 178 standard. However, Ne 178 does not provide specific guidance on how each VoR step should be implemented. Therefore, their implementation is one of the main focuses of this paper, which will be discussed in detail in Section 5. The functionalities of the VoR steps, in the order of their execution, are described below.

- Firstly, the central M+O server, located in the IT domain (e.g., a cloud), generates optimal setting points based on real-time process data using a process optimization algorithm. As the request issuer, it initiates a CPC reconfiguration request to VoR Step 1: Authentication and authorization, situated within the interface partition. Upon receiving the request, VoR Step 1 verifies the trustworthiness and control reconfiguration authority of the request issuer by assessing its credential.
- Following successful authentication and authorization, the request is forwarded to VoR Step 2: Request verification, where the message completeness of the request is validated to ensure it contains all necessary core components, such as modification information (cf. Fig. 2).
- The complete request is then passed to VoR Step 3: Mapping, where a mapping algorithm processes the request data and generates a list of affected endpoints, along with mapping verification constraints (cf. Appendix A3), which are forwarded to the next VoR step.
- In the VoR Step 4: Propagation, the mapped request is stored in a priority queue, awaiting retrieval by the propagation frontend.
- VoR Step 5: Acceptance periodically monitors the status of the process control system to determine whether control reconfiguration is feasible. Once the system is ready, the request retrieved by the propagation frontend from the priority queue will undergo verification process in VoR Step 6.

- In this final VoR Step 6: Mapping verification, the effective setpoints for the affected endpoints are calculated based on the modification information given in the request, such as a relative change of "30%." Once all effective setpoints have been computed, they are verified against the mapping verification constraints specified by VoR Step 3, for example, to ensure that the values fall within allowed ranges. After successful mapping verification, the effective setpoints for all affected endpoints along with their corresponding endpoint IDs are sent in their entirety to the client within the data aggregation system. The effective values to be updated are then written by the client to the process control server, completing the control reconfiguration process.

It is important to note that, regardless of how a request is processed across the different VoR steps, the processing results associated with the request, such as the mapped list of affected endpoints, must be transmitted in their entirety. Partial processing or execution of a request is prohibited. Additionally, if a request fails to pass the verification at any given VoR step, it will not be forwarded to subsequent VoR steps for further verification.

5. System implementation

Building on our previously proposed architecture (cf. Section 4.2), this section provides a detailed description of the proposed system's implementation.

The implementation was carried out within a Python environment, leveraging its extensive ecosystem for rapid development and integration.

It is worth mentioning that, to help the reader understand the purpose of each VoR step, seven specific questions and one action reflecting the requirements of NE 178 are presented at the beginning of each step's implementation description and addressed in the corresponding subsections.

5.1. VoR system implementation.

5.1.1. Authentication and authorization. Through this step, the following two questions will be answered:

- *Who is the request issuer?*
- *What is this request issuer allowed to do?*

Users or process optimization applications within the central M+O domain send requests for control reconfiguration. The OPC UA was selected for its proven value across various industrial automation domains and was therefore integrated into the VoR system proposed in this work. The requests are sent via OPC UA method calls from an OPC UA client to the OPC UA server located in the interface partition of the SIoT edge gateway. The request payload passed in the OPC UA method call is shown in Fig. 2. The server authenticates the OPC UA client using X.509 v3 public key certificates associated with a corresponding private key. The issuer is authorized using credentials in the form of a username and password tuple.

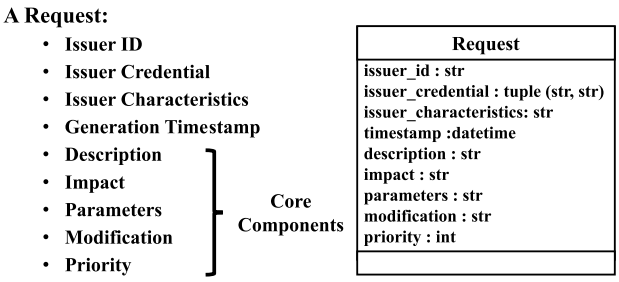

Fig. 2. Component of a request and the corresponding request class.

Once authenticated, the requested modifications to the given parameters (e.g., equipment information) are analyzed to determine whether the request issuer is authorized. To this end, a role-based access control (RBAC) model is employed, mapping valid issuer credentials to roles (e.g., administrator, plant operator, maintenance personnel) and associating parameters with them. This is implemented using the Python library pyCasbin, in which the access control model is defined in a configuration file (cf. Appendix A1). This setup enables detailed modeling of the authorization process and allows the authorization logic to be updated throughout the system's lifecycle. In the case of successful authorization, the request is forwarded via POSIX MQ to VoR Step 2 for request verification.

5.1.2. Request verification. Through this step, the following two questions will be answered:

- *Is the current request complete?*
- *Are the inputs provided in the request valid?*

After receiving the request from VoR Step 1, this step verifies the message completeness of the request. Specifically, the XML-formatted request data passed from VoR Step 1 is first instantiated as an object of the request class (cf. Fig. 2). By using the request class as a unified template to instantiate all incoming requests, a consistent format is enforced across the system. This not only ensures uniform parsing, validation, and handling of requests across all VoR components, but also facilitates debugging, monitoring, and auditing, thereby enhancing the maintainability of the system.

Subsequently, request verification is carried out by checking each rule listed in the XML-encoded request verification rule set to verify the message completeness and message validity of the current request, i.e., whether the core components of the request have been provided completely and correctly by the request issuer. The rules in the request verification rule set are all in Boolean format. A request will only pass the verification if it returns true for all the rules. If any single rule returns false, the request will be rejected, and the description of that rule will be used as feedback in the VoR system's log.

To give the reader a more concrete understanding of request verification, an example of an XML-encoded request verification rule set is provided in Appendix A2. It is important to note that the configuration of that set is determined solely by the specific application scenario, allowing users to edit the rules according to their particular application needs.

5.1.3. Mapping. Through this step, the following question will be answered:

- *Which CPC parameter values are to be modified?*

In the context where information of OT systems cannot be exposed to the IT domain, the mapping step determines how the model parameters and changes specified in the request correspond to specific data objects, such as process or control parameters, within CPC. In this step, mapping is performed by matching the "Impact" field of the incoming request with the names of rule sets defined in the mapping rule sets. This determines the intended type of control reconfiguration and translates the model change described in the request into a list of ordered parameter modifications for the OT project, referred to as the list of affected endpoints. For illustrative purposes, an example of XML-encoded mapping rule sets is provided in Appendix A3. The included mapping rule sets contain three representative cases:

- *Single-parameter reconfiguration.* In this case, the affected endpoint is singular, such as when opening/closing a valve or adjusting motor speed, corresponding to the "Motor speed configuration" rule set in Appendix A3.
- *Multi-parameter reconfiguration.* In such cases, multiple interrelated parameters need to be reconfigured simultaneously. For instance, increasing the feed rate of material A while maintaining a fixed production recipe requires proportionally increasing the feed rates of other materials, corresponding to the "Flow rate adjustment" rule set in Appendix A3.
- *MTP-based multi-parameter reconfiguration.* In MTP-based modular plants, changes to the process control of a modular unit must be performed via state machine transitions. Even if the intended modification relates to only a single parameter, the system must first be switched to a state that permits control reconfiguration (such as idle state) before parameter adjustments can be made, corresponding to the "Starting stirring" rule set in Appendix A3.

Although MTP-based reconfiguration is more intricate, it enables vendor-independent modular control by utilizing standardized MTP-defined parameters, eliminating the need for internal OT-specific details, such as user-defined parameters. Additionally, the rich set of MTP parameters supports more granular control reconfiguration possibilities, for example, allowing not only service-level reconfigurations but also deeper control reconfigurations at the procedure level.

Only control reconfiguration types that are pre-defined in the mapping rule sets can be mapped to the corresponding list of affected endpoints. If the VoR mapping step cannot find a corresponding mapping rule set for the "Impact" information in the request within the mapping rule sets, the request is considered invalid and is subsequently rejected. Similar to the request verification rule set, the configuration of the mapping rule sets is determined solely by the specific application scenario, allowing users to edit the rules according to their particular application needs.

5.1.4. Propagation. Through this step, the following action will be carried out:

- *Let me store the request in a priority queue and hold it until it is called!*

The functionality of the VoR propagation step is achieved through two components: the propagation backend and frontend. In the proposed VoR system, the former is responsible for running a priority message queue to store the mapped requests before they are pulled by the frontend. Since the VoR system is designed to operate on an edge device, resource efficiency is a key consideration when selecting the technology. Furthermore, in the event of unexpected situations, such as power outages, it is crucial that mapped requests not be lost. Therefore, the ability to persistently store mapped requests locally is another important factor in the selection of the technology. Additionally, for an incremental set of requests, enabling their continuous execution can be achieved by setting their priority to the same level. Consequently, the chosen technology must support priority queues. In contrast, concurrency management is not a significant concern in the use case studied in this work, as only the propagation frontend pulls requests from the message queue, meaning there is no scenario involving multiple message consumers. Based on this analysis, SQLite is

selected as the technology to implement the propagation backend, since it provides a serverless, lightweight solution that supports local persistent storage and priority queues.

The propagation frontend is responsible for retrieving the next highest-priority request from SQLite after the completion of the last VoR step for further processing. The timing of this retrieval depends on the completion status of the last step (mapping verification).

5.1.5. Acceptance. Through this step, the following question will be answered:

- *Is the OT device now ready for configuration?*

Since VoR Step 5 (acceptance) involves verifying the operational state of OT devices, which requires retrieving real-time data of the plant control status from the OT domain via the data aggregation system, a communication interface between VoR Step 5 and the data aggregation system must be established. Considering lightweight implementation, XML data compatibility, and future interoperability among containerized services, a RESTful API built with Flask was selected as the communication mechanism.

A REST client is deployed within the VoR propagation frontend, while the data aggregation system hosts a corresponding Flask server. The REST client posts XML-encoded requests to the Flask server to query relevant information such as the current operation mode of the OT equipment, warning states, and the process values of affected endpoints. Upon receiving the request, the Flask server generates and returns the appropriate XML-encoded payload in response. Once the required data is received, VoR Step 5 proceeds to determine whether the OT system is in a valid state to perform the requested control reconfiguration.

Similar to the request verification rule set and the mapping rule sets, the acceptance step also requires a rule set that includes the parameters to be verified and their required values, in order to determine the current availability of the OT device for control reconfiguration. An example of an acceptance rule set is provided in Appendix A4. Similarly, the configuration of the acceptance rule set is determined solely by the specific application scenario, allowing users to edit the rules according to their particular application needs.

5.1.6. Mapping verification. Through this step, the following question will be answered:

- *Are the parameter values to be modified plausible?*

VoR Step 6 (mapping verification) is the final step of the VoR system, responsible for verifying data validity. By combining the current process values of the affected endpoints obtained from the REST client with the modification descriptions of the request, mapping verification first calculates the effective value to be set for each affected endpoint. Then, it uses the mapping verification constraints provided in VoR Step 3 to validate each effective value. If all the effective values pass the verification, the values of all affected endpoints, along with their corresponding affected endpoint IDs, will be sent to the OPC UA client running in the data aggregation system. The client then writes the corresponding effective values to the proper nodes on the process control server running on the OT devices, ultimately completing the control reconfiguration process.

5.2. Data aggregation system. The data from the CPC system required for VoR Step 5 (acceptance) is gathered using an aggregation system. It functions firstly as a protocol bridge supporting the OPC UA and ModbusTCP servers in the OT domain. For this, the system implements concurrent client interfaces that establish connections with industrial control systems via the respective protocols. Secondly, it serves as a data aggregator by aggregating specified data endpoints frequently into its data pool. Data acquisition occurs through configurable periodic read operations defined in XML configuration files that specify server information (e.g., IP address, port number, security policies) endpoints, node identifiers, and sampling intervals. Examples of these configuration files are given in Appendices B1 and B2.

The aggregation system's architecture employs multi-threading to manage these distinct communication channels simultaneously, with dedicated threads for each protocol type and sequential calls to the protocol clients. The aggregation system exposes real-time data through a Flask-powered RESTful API. This API enables the VoR acceptance step to query operational states, warning conditions, and process values on demand. The Flask server handles incoming requests by dynamically generating XML payloads containing the requested data subsets.

5.3. Containerization. As shown in Fig. 1, the implemented VoR system consists of five deployed VoR step applications, namely, VoR_1, VoR_2, VoR_3, VoR_4, and data aggregation. Due to their functional and operational independence, each of these applications is deployed and executed within its own Docker container. This enhances modularity, flexibility, and fault isolation, which ensures independent operation of components, simplifying maintenance, testing, and updates. Containers provide a consistent environment, support resource isolation, and optimize resource usage, making the system scalable and portable across edge platforms. This setup ensures reliable operation and easier deployment across

various industrial scenarios.

5.4. XML generator for a rule set. Considering the compatibility of the OPC UA with XML formatted data, XML is chosen as the format for data transmission between VoR steps and for rule sets within the VoR system, enhancing compatibility with the OPC UA and ensuring internal consistency within the VoR system.

To simplify the process of creating XML-encoded rule sets for VoR Steps 2, 3, 5 and ensure their correctness, we provide users with an XML generation tool, which serves as a user interface (UI) for rule set creation. The tool generates XML that adheres to the NE 178 standard (NAMUR, 2025), ensuring the validity and consistency of the rule sets. It also allows users to load and modify an existing rule set, eliminating the need to create a new one from scratch each time.

5.5. Implementation diagram of the VoR system. The technical details of the implemented VoR system are depicted using a bottom-up diagram of process data flow (Fig. 3) and a top-down diagram of VoR step execution (Fig. 4).

6. Research method

This study follows the experimentation framework proposed by Basili *et al*. (1986) and adopts Shaw's classification of software engineering validation approaches (Shaw, 2003). The research design consists of a structured study definition and a laboratory-based functional evaluation.

6.1. Study definition. The VoR mechanism specified in NE178 has not yet been fully realized in industrial practice. Motivated by this gap, the study aims to implement and evaluate the proposed VoR system within a realistic NOA context. The object of study is the VoR system responsible for verifying requests transferred from the IT domain to the OT one. The purpose is to characterize and evaluate the correctness, completeness, functional behavior, and practical applicability of the established VoR system. The perspective is that of industrial automation engineers, NOA integrators, and system architects, who require standardized, traceable, and scalable request verification workflows. The domain encompasses NOA request messages and verification procedures defined by NE 178, and the scope is limited to a laboratory evaluation of a single-hardware VoR implementation, in which the VoR module performs realistic module reconfiguration operations, and a systematically defined set of request cases, including both valid and invalid requests, is used to verify its correctness and functional behavior.

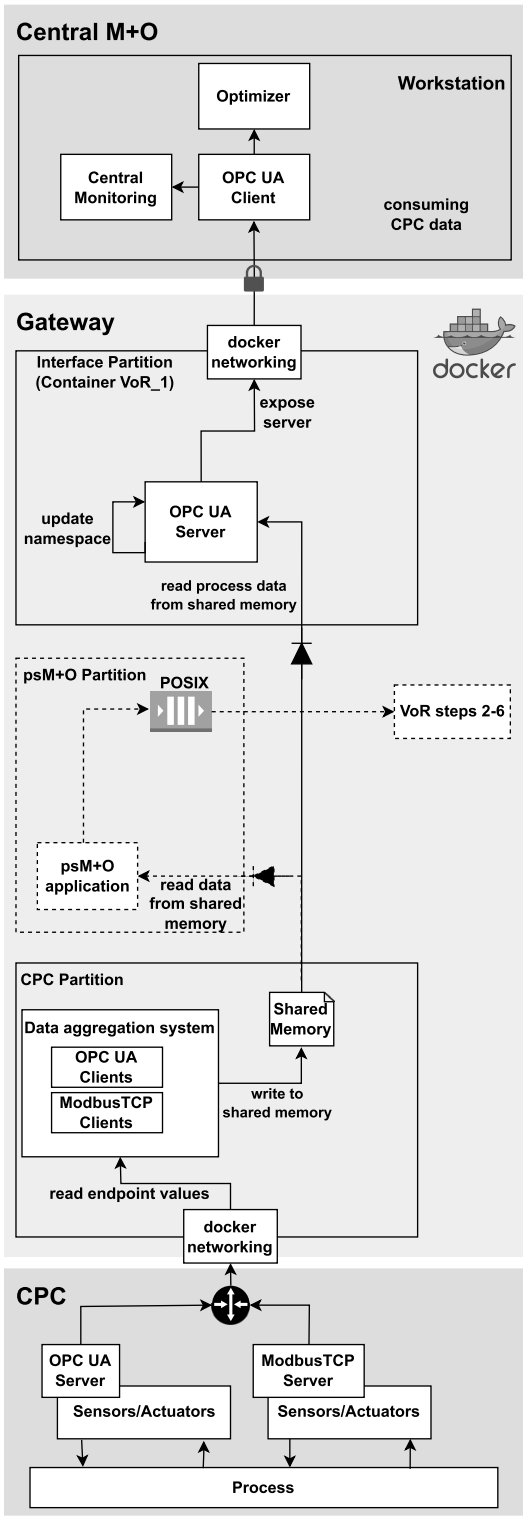

Fig. 3. Bottom-up diagram of process data flow.

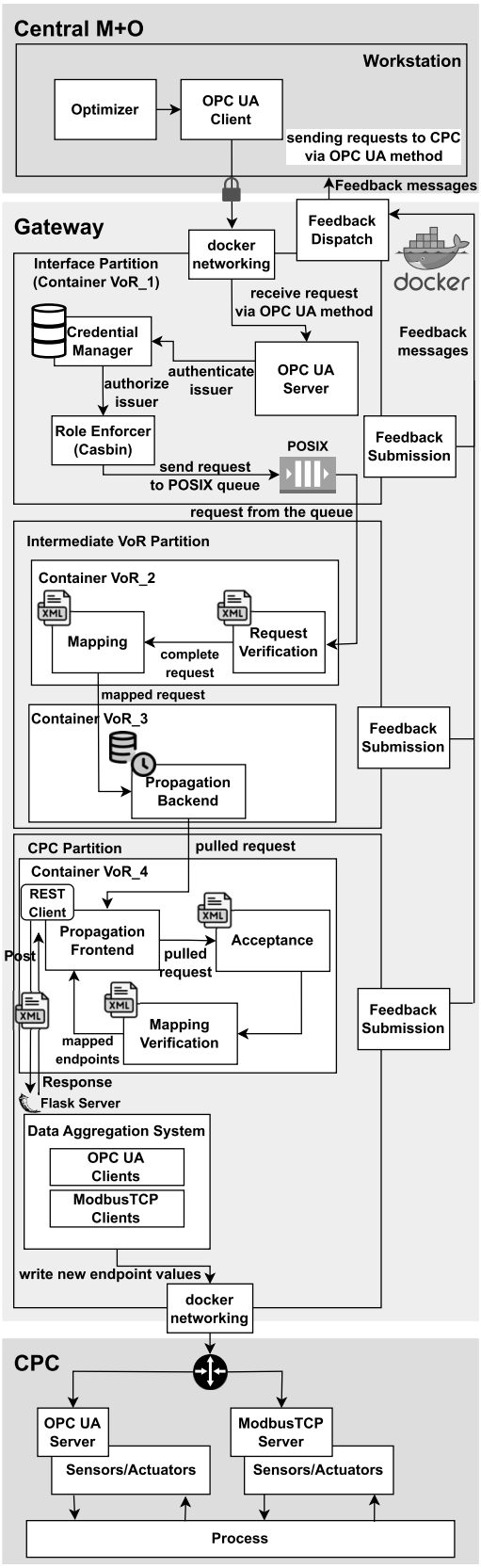

Fig. 4. Top-down diagram of VoR step execution.

6.2. Evaluation method. To address RQ1–RQ4 from Section 2, a mixed evaluation approach was employed, reflecting Shaw's classification of validation methods: analytical, example-based, and experience-based evaluations, as discussed below.

1. *Analytical evaluation (RQ1: Architectural completeness and feasibility).* The implemented verification logic was examined against all mandatory NE 178 verification functions to ensure completeness at the policy, structural, and semantic levels. This evaluation directly addresses RQ1 by assessing whether a single-hardware VoR architecture can fully implement authentication and authorization functions, structural-integrity checks, semantic-mapping validation, and permission and parameter-constraint verification.

2. *Example-based evaluation (RQ2: Verification accuracy and functional behavior).* Eight test cases were designed to systematically cover all verification categories required by NE 178, including one valid and seven invalid requests (cf. Table 2), each targeting a specific verification function. The established VoR system was expected to accept the valid request and reject the invalid ones, and provide detailed, traceable feedback. This evaluation demonstrates the mechanism's accuracy and functional behavior, directly addressing RQ2.

3. *Experience-based evaluation (RQ3 and RQ4: Cross-domain applicability, reusability, and modular plant reconfiguration).* A controlled NOA laboratory environment was used to evaluate the VoR system during realistic module reconfiguration scenarios. This experience-based evaluation demonstrates that the VoR mechanism can be implemented using a single hardware component and achieves cross-domain adaptability and reusability through a configurable rule set. The rule set can be generated and modified using the XML generator described in Section 5.4, allowing users to tailor the configuration to their specific application requirements. Furthermore, the proposed VoR system can incorporate modular plant concepts such as the MTP standard (cf. Section 5.1.3), thereby addressing RQ3 and RQ4.

This methodological design ensures that each research question is supported by appropriate evidence and aligns with established software engineering practices.

7. Validation

The proposed VoR system has been implemented on an industrial PC (IPC). The structure of the demonstrator is illustrated in Fig. 5. The IPC was connected to the process

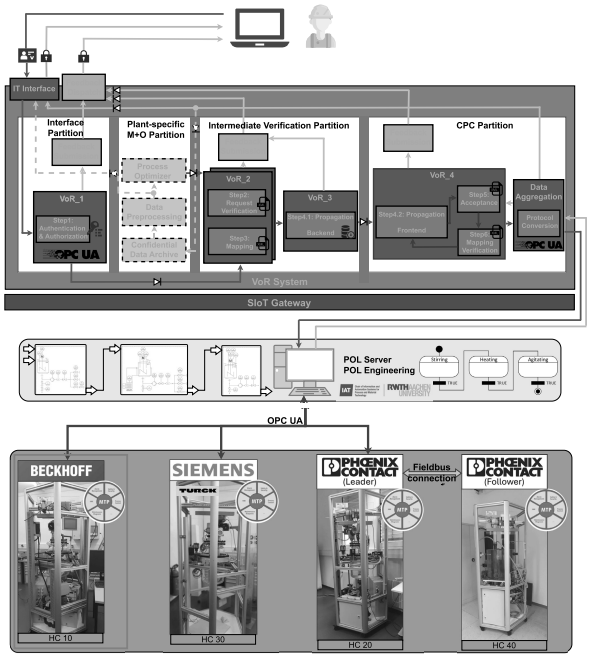

Fig. 5. Structure diagram of the demonstrator.

orchestration layer (POL) server via an Ethernet cable. Communication between the IPC and the POL server was established using the OPC UA over HTTP protocol, where the POL server hosted an OPC UA server while the data aggregation system deployed on the IPC hosted an OPC UA client.

Since cloud deployment is still in progress, the test cases, including one sample valid test case and seven sample invalid test ones (in order: unauthenticated access, unauthorized configuration, malformed request, undefined mapping handling, null request handling, permission constraint violation, variable value range violation) for verifying the established VoR system's function, listed in Table 2, were conducted in a laboratory setting, where the optimization request was manually triggered by plant operators ("YZ": the administrator, "Alice": the regular operator) via the OPC UA method *request* (cf. Fig. 6).

The executed test cases successfully validated the proposed VoR system's ability to handle control reconfiguration tasks in modular plants (cf. Table 2). For the valid test case "Motor speed configuration to 700 rpm (revolutions per minute) in HC10", the VoR system verified the request, correctly mapped it to the affected endpoint *HC10_MotorSpeed_SP*, and reconfigured its value to 700.0. For the invalid test cases, the attempted control reconfigurations were not executed, and the corresponding feedback messages were generated and logged by the established VoR system. This demonstrated the established VoR system's functional correctness and reliability in a laboratory environment.

```
Input Arguments
   IssuerID: String = "YZ"
   Credentials: String = "
       Admin123_secure_password_2025"
   IssuerCharacteristics: String = "{IP
       address of the Request Issuer:
       192.168.0.0}"
   GenerationTimestamp: DateTime =
       "2025-10-09T19:00:00.000Z"
   Description: String = "{'Change the
       motor speed', 'in HC10'}"
   Impact: String = "motor speed
       configuration"
   Parameters: String = "HC10"
   Modification: String = "700"
   Priority: Int32 = 3

Output Arguments
   RequestID: String = "818fa573-0f96-4
       aca-ab75-26e92d943a1e"
   Timestamp: String = "2025-10-09T19
       :13:28.313343+00:00"
   Notification: String = "Submission
       received"

Result
   Status: Succeeded
```

Fig. 6. Sample input and the corresponding output of a valid request via the OPC UA method *request*.

8. Conclusion and outlook

This study presents an edge-native VoR system designed to enable secure and plausible control reconfiguration in modular plant environments. Building upon the architecture proposed in our previous work, this paper provides a comprehensive account of the implementation of each step within the VoR system, and explains the rationale behind the selection of technologies, thereby addressing a significant gap in practical VoR implementations and their adoption in the field. Furthermore, particular emphasis is placed on ensuring the plausibility, integrity, and scalability of control reconfiguration processes in modular plants. The implementation was successfully tested using the modular automation unit HC10, where both valid and invalid remote control configuration requests were processed. Valid requests were executed with full reliability and correctness, while invalid ones triggered corresponding feedback messages that were generated and logged by the VoR system, demonstrating its effectiveness in a realistic industrial laboratory environment.

Several important lessons were learned during the development and evaluation of the VoR system. They indicate that a standardized and consistently formatted NOA request model, full integration of

Table 2. Sample test cases for VoR functionality conducted in the laboratory setting.

Tested function	Test description	VoR feedback message	Test status
Valid test case	A valid request configured the stirring motor speed to 700 rpm in HC10.	"Success: {'HC10_MotorSpeed_SP', 'value': 700.0, 'status': 'done'}"	✓
VoR Step 1: Authentication	Unauthenticated personnel are not allowed to submit requests.	"VoR1: Authentication failed for 'YZ_unathenticated'"	✓
VoR Step 1: Authorization	Unauthorized personnel (due to lack of permissions) are not allowed to configure the corresponding variables.	"VoR1: Authorization failed for 'Alice'"	✓
VoR Step 2: Request verification	Incomplete requests will be rejected (in this test case, due to a missing request description).	"VoR2: A specific rule has disapproved the request based on its contents. Failed Rules: ['1']"	✓
VoR Step 3: Mapping	Undefined mappings are not permitted to execute.	"VoR3: Mapping failed: no matching mapping rule set"	✓
VoR Step 4: Propagation	The propagation frontend attempts to pull a request even when none is registered in the database.	"VoR4: No records found in the database"	✓
VoR Step 5: Acceptance	Devices failing control reconfiguration requirements cannot be reconfigured.	"VoR5: The request is not accepted: key_personnel_present, current='False'"	✓
VoR Step 6: Mapping verification	Variable values outside the allowed range are not permitted to be written.	"VoR6: Not completed: The request failed the mapping verification."	✓

verification into the operational workflow, structured error classification with corresponding traceable feedback, and a combination of formal constraints with rule-based and example-based testing are critical for reliable IT/OT request verification. The validation results showed that a modular, rule-based design significantly simplifies VoR system integration and maintenance while supporting cross-domain adaptability through configurable rule sets. Close collaboration between control engineers and IT developers was essential to ensure operational safety while maintaining system openness, and continuous testing and modularization improved efficiency and system reliability. Taken together, these insights and the VoR system implementation provide practical guidance for researchers and practitioners and serve as a reference model for future NOA software implementations and other industrial contexts.

Future work will focus on integrating the VoR system with cloud-based M+O applications and establishing a closed-loop mechanism for adaptive control reconfiguration in modular plants. Efforts will also be directed toward optimizing the maintenance and upgrade processes to enable flexible deployment of VoR applications. Finally, the proposed system will be used in real-world industrial environments, where continuous feedback will facilitate iterative improvements and further optimization of system performance.

Acknowledgment

The research leading to these results had been funded by the German Federal Ministry of Research, Technology and Space (BMFTR) under the grant agreement no. 16KIS1962 (SIoT-Gateway).

References

Aro, J. (2021). OPC UA PubSub explained, `https://prosysopc.com/blog/opc-ua-pubsub-explained/`.

Azarmipour, M., Trotha, C.v., Gries, C., Kleinert, T. and Epple, U. (2020a). A secure gateway for the cooperation of information technologies and industrial automation systems, *46th Annual Conference of the IEEE Industrial Electronics Society IECON 2020, Singapore,* pp. 53–58, DOI: 10.1109/IECON43393.2020.9254634.

Azarmipour, M., von Trotha, C., Epple, U., Ansar, Z. and Gries, C. (2020b). Realisierung der NAMUR-Diode mittels Virtualisierung, *atp magazin* **62**(5): 62–67, DOI: 10.17560/atp.v62i5.2472.

Baldea, M., Edgar, T.F., Stanley, B.L. and Kiss, A.A. (2017). Modular manufacturing processes: Status, challenges, and opportunities, *AIChE Journal* **63**(10): 4262–4272, DOI: 10.1002/aic.15872.

Basili, V.R., Selby, R.W. and Hutchens, D.H. (1986). Experimentation in software engineering, *IEEE Transactions on Software Engineering* **SE-12**(7): 733–743, DOI: 10.1109/TSE.1986.6312975.

Boyes, H., Hallaq, B., Cunningham, J. and Watson, T. (2018). The industrial Internet of things (IIoT): An analysis framework, *Computers in Industry* **101**: 1–12, DOI: 10.1016/j.compind.2018.04.015. .

FOIS (2022). OPC UA security analysis, *Technical report*, Federal Office for Information Security, Bonn, https://opcfoundation.org/wp-content/uploads/2023/11/BSI-OPCUA-2022-EN.pdf.

Grüner, S. and Trosten, A. (2023). A cloud-native software architecture of NAMUR Open Architecture verification of request using OPC UA PubSub actions over MQTT, *2023 IEEE 28th International Conference on Emerging Technologies and Factory Automation (ETFA), Sinaia, Romania*, pp. 1–8, DOI: 10.1109/ETFA54631.2023.10275714.

Hollerer, S., Kastner, W. and Sauter, T. (2021). Towards a threat modeling approach addressing security and safety in OT environments, *2021 17th IEEE International Conference on Factory Communication Systems (WFCS), Linz, Austria,* pp. 37–40, DOI: 10.1109/WFCS46889.2021.9483591.

Honda, T., Hamaguchi, T. and Hashimoto, Y. (2022). OPC UA information transfer via unidirectional data diode for ICS cyber security, *14th International Symposium on Process Systems Engineering, Kyoto, Japan*, pp. 1459–1464, DOI: 10.1016/B978-0-323-85159-6.50243-8.

Maharjan, R., Chy, M.S.H., Arju, M.A. and Cerny, T. (2023). Benchmarking message queues, *Telecom* **4**(2): 298–312, DOI: 10.3390/telecom4020018.

NAMUR (2020). *NAMUR Recommendation NE 175: NAMUR Open Architecture: NOA Concept*, NAMUR, Leverkusen, https://www.namur.net/en/publications/news-archive/ne-175-is-newly-published.html.

NAMUR (2025). *NAMUR Recommendation NE 178: NAMUR Open Architecture: Verification of Request*, NAMUR, Leverkusen, https://www.namur.net/en/publications/news-archive/ne178-namur-open-architecture-verification-of-request-is-newly-published.html.

OPC Foundation (2023). UA Part 2: Security, *Specification OPC 10000-2*, OPC Foundation, Scottsdale, https://reference.opcfoundation.org/specs/OPC-10000-2.

OPC Foundation (2024). UA Part 14: PubSub, *Specification OPC 10000-14*, OPC Foundation, Scottsdale, https://reference.opcfoundation.org/specs/OPC-10000-14.

Pakala, H.K. and Diedrich, C. (2025). Modelling of the NAMUR NE 178 "Verification of Request" concept using Pro-Active Asset Administration Shells, *at Automatisierungstechnik* **73**(2): 114–124, DOI: 10.1515/auto-2024-0141.

Ryu, D., Lee, S., Yang, S., Jeong, J., Lee, Y. and Shin, D. (2024). Enhancing cybersecurity in energy IT infrastructure through a layered defense approach to major malware threats, *Applied Sciences* **14**(22): 10342, DOI: 10.3390/app142210342.

Schmetz, B., Winter, M., Miny, T. and Kleinert, T. (2025). MTP-Realisierung an einer Multi-Vendor Anlage, *ATP Magazine* **65**(5): 46–54, DOI: 10.17560/atp.v67i5.2779.

Shaw, M. (2003). Writing good software engineering research papers, *25th International Conference on Software Engineering, Portland, USA,* pp. 726–736, DOI: 10.1109/ICSE.2003.1201262.

Zhao, Y., Shagufta, S. and Kleinert, T. (2024). Flexible control configuration in modular plants via control function library, *2024 IEEE 29th International Conference on Emerging Technologies and Factory Automation (ETFA), Padova, Italy*, pp. 1–7, DOI: 10.1109/ETFA61755.2024.10710925.

Zhao, Y., Schoon, H., Schmetz, B., Miny, T., Felke, P. and Kleinert, T. (2025). An architecture of a NOA-based secure Internet of things edge gateway, *2025 IEEE 8th International Conference on Industrial Cyber-Physical Systems (ICPS), Emden, Germany,* pp. 1–6, DOI: 10.1109/ICPS65515.2025.11087862.

Yuanchen Zhao received his MSc degree in automation engineering from RWTH Aachen University in 2022. He subsequently began his doctoral studies in the field of flexible process automation and control at the Chair of Information and Automation Systems for Process and Material Technology at RWTH Aachen University. Since 2024, he has been a member of the joint ZVEI/NAMUR working group WG 2.8.4 *Automation Architecture—NOA Verification of Request*. His research interests include distributed cloud-edge systems for flexible process automation, industrial cybersecurity, and industrial compiler construction.

Heiko Schoon received his BEng degree in electrical engineering from the University of Applied Sciences Emden/Leer in 2023. He is currently pursuing his MEng degree in industrial informatics, with a focus on industrial cyber-physical systems, engineering of Industry 4.0-capable solutions, robotics, and automation. He is also working as a research assistant at the Chair of IT Security at the University of Applied Sciences Emden/Leer. His research interests include cyber-physical systems, industrial cybersecurity, and secure automation technologies.

Torben Miny received is MSc and PhD degrees in automation from RWTH Aachen University, Germany, in 2016 and 2022, respectively. Since 2022, he has been a senior researcher with the Chair of Information and Automation Systems for the Process and Material Technology, RWTH Aachen University. His area of expertise lies in information modeling and the automated semantic exchange of this information in the Industry 4.0 environment. In this context, he deals with the topics of asset administration shell, discovery, data integration and data spaces.

Patrick Felke received his MSc degree in mathematics from TU Dortmund University, Germany, in 2001, and his PhD degree in mathematics from Ruhr University Bochum, Germany, in 2005. He is currently a professor of IT security with the University of Applied Sciences Emden/Leer, Germany. His research and teaching interests are cryptology, protocol security and hacking.

Tobias Kleinert is a graduate in mechanical engineering from RWTH Aachen University and holds a PhD from Ruhr University Bochum. He has worked at BASF SE in various fields of industrial automation and digitalisation. Since 2020, he has been the head of the Chair of Information and Automation Systems for Process and Material Technology at RWTH Aachen University. His focus of teaching and research is industrial data systems as well as flexible industrial automation and control.

Appendix A

A sample role-based access control list and corresponding rule sets for the proposed VoR system described in this paper are presented below.

A1. Role-based access control list

```
// Admin has full access to all resources
p, Admin, HC10, motor speed configuration
p, Admin, HC10, flow rate adjustment
p, Admin, HC20, motor speed configuration
p, Admin, HC20, flow rate adjustment

// PlantOperator can operate basic machinery
p, Operator_HC10, HC10, flow rate adjustment

// Define user-role assignments (g)
g, YZ, Admin
g, Alice, Operator_HC10
g, Bob, Operator_HC10
g, Cindy, Operator_HC20
```

A2. Request verification rule set

```
<ruleset>
    <rule id="1">
        <description>
            Request description can't be empty
        </description>
        <condition>
            bool(request.description)
        </condition>
    </rule>
    <rule id="2">
        <description>
            Priority must be a positive integer
        </description>
        <condition>
            Request.priority &gt; 0
        </condition>
    </rule>
    <rule id="3">
        <description>
            Request must contain at least one parameter
        </description>
        <condition>
            bool(request.parameters)
        </condition>
    </rule>
    <rule id="4">
        <description>
            Modification field must not be empty
        </description>
        <condition>
            bool(request.modification)
        </condition>
    </rule>
</ruleset>
```

A3. Mapping rule sets

```
<MappingRuleSets>
  <MappingRuleSet name="motor speed configuration">
    <Rule id="Rule_1">
      <TriggerCondition>
        Set Motor Speed
      </TriggerCondition>
      <ChangeDescription>
        Configure Motor Speed
      </ChangeDescription>
      <EndpointIdentifier>
        MotorSpeed_SP
      </EndpointIdentifier>
      <UnitOfChange>
        RPM
      </UnitOfChange>
      <MappingVerificationConstraint>
        &lt;= 2000
      </MappingVerificationConstraint>
    </Rule>
  </MappingRuleSet>

  <MappingRuleSet name="flow rate adjustment">
    <Rule id="Rule_1">
      <TriggerCondition>
        Need to increase flow rate of Material A
      </TriggerCondition>
      <ChangeDescription>
        Change the pump's power
      </ChangeDescription>
      <EndpointIdentifier>
        Pump_A_Power
      </EndpointIdentifier>
      <UnitOfChange>
        kw
      </UnitOfChange>
      <MappingVerificationConstraint>
        &lt; 20
      </MappingVerificationConstraint>
    </Rule>
    <Rule id="Rule_2">
      <TriggerCondition>
        Maintain A:B = 1:3
      </TriggerCondition>
      <ChangeDescription>
        Change the pump's power
      </ChangeDescription>
      <EndpointIdentifier>
        Pump_B_Power
      </EndpointIdentifier>
      <UnitOfChange>
        kw
      </UnitOfChange>
      <MappingVerificationConstraint>
        &lt; 30
      </MappingVerificationConstraint>
    </Rule>
  </MappingRuleSet>

  <MappingRuleSet name="starting stirring">
    <Rule id="Rule_1">
      <TriggerCondition>
        Start Stirring Motor
      </TriggerCondition>
      <ChangeDescription>
        Activate operator mode
      </ChangeDescription>
      <EndpointIdentifier>
        StateOpOp
      </EndpointIdentifier>
```

```
      <UnitOfChange>
        unitless
      </UnitOfChange>
      <MappingVerificationConstraint>
        none
      </MappingVerificationConstraint>
    </Rule>
    <Rule id="Rule_2">
      <TriggerCondition>
        Start Stirring Motor
      </TriggerCondition>
      <ChangeDescription>
        start motor
      </ChangeDescription>
      <EndpointIdentifier>
        FwdEn
      </EndpointIdentifier>
      <UnitOfChange>
        unitless
      </UnitOfChange>
      <MappingVerificationConstraint>
        none
      </MappingVerificationConstraint>
    </Rule>
</MappingRuleSets>
```

A4. Acceptance rule set

```
<ruleset>
    <key_personnel_present>
        <current_value>none</current_value>
        <required_value>true</required_value>
    </key_personnel_present>

    <technical_system id="System_1">
        <availability>
            <current_value>none</current_value>
            <required_value>available</required_value>
        </availability>
        <operation_mode>
            <current_value>none</current_value>
            <required_value>idle</required_value>
        </operation_mode>
        <warning>
            <current_value>none</current_value>
            <required_value>no warnings</required_value>
        </warning>
    </technical_system>
</ruleset>
```

Appendix B

Sample configuration files that define the OT data endpoints for the data aggregation system are presented

B1. OPC UA endpoints

```
<server>
  <server_app_uri>
    opc.tcp://brownfield-opcua-server:4840/freeopcua/server/
  </server_app_uri>
  <client_app_uri>
    urn:Client:foobar:docker-client
  </client_app_uri>
  <alias>secured-brownfield-opcua-server</alias>

  <security>
    <policy>SecurityPolicyAes256Sha256RsaPss</policy>
    <mode>SignAndEncrypt</mode>
    <server_certificate>
      certificates/trusted/brownfield_server_cert.der
    </server_certificate>
    <client_certificate>
      certificates/client_cert.der
    </client_certificate>
    <client_private_key>
      certificates/client_key.pem
    </client_private_key>
  </security>

  <nodes>
    <node>
      <DisplayName>KeyPersonnelPresent</DisplayName>
      <NamespaceIndex>0</NamespaceIndex>
      <IdentifierType>Numeric</IdentifierType>
      <Identifier>20004</Identifier>
      <datatype>Boolean</datatype>
      <description>Key personnel is present</description>
    </node>

    <node>
      <DisplayName>SystemAvailable</DisplayName>
      <NamespaceIndex>0</NamespaceIndex>
      <IdentifierType>Numeric</IdentifierType>
      <Identifier>20005</Identifier>
      <datatype>Boolean</datatype>
      <description>System1 is available</description>
    </node>

    <node>
      <DisplayName>OperationMode</DisplayName>
      <NamespaceIndex>0</NamespaceIndex>
      <IdentifierType>Numeric</IdentifierType>
      <Identifier>20006</Identifier>
      <datatype>String</datatype>
      <description>Current operation mode</description>
    </node>

    <node>
      <DisplayName>Warning</DisplayName>
      <NamespaceIndex>0</NamespaceIndex>
      <IdentifierType>Numeric</IdentifierType>
      <Identifier>20007</Identifier>
      <datatype>String</datatype>
      <description>Warning state of System1</description>
    </node>
  </nodes>
</server>
```

B2. ModbusTCP endpoints

```
<server>
  <ipaddr>172.20.128.2</ipaddr>
  <port>503</port>
  <serveralias>Ecotec</serveralias>
  <endpoints>
  <endpoint>
    <name>M7 Failure Condensat</name>
    <function>Read Coils</function>
    <address>8262</address>
    <quantity>1</quantity>
    <offset>-1</offset>
    <type>Coils</type>
    <description>M7 Failure in Condensat</description>
  </endpoint>

  <endpoint>
    <name>M10 Failure Compressor 2</name>
    <function>Read Coils</function>
    <address>8265</address>
    <quantity>1</quantity>
    <offset>-1</offset>
    <type>Coils</type>
    <descriptionM10 Failure Compressor 2</description>
  </endpoint>
  <endpoint>
    <name>M11 Failure Compressor 3</name>
    <function>Read Coils</function>
    <address>8266</address>
    <quantity>1</quantity>
    <offset>-1</offset>
    <type>Coils</type>
    <description>M11 Failure Compressor 3</description>
  </endpoint>
</server>
```

Received: 1 May 2025
Accepted: 5 February 2026