Enhanced Simulation Environment to Support Research in Modular Manufacturing Systems

Björn Leander*†, Tijana Marković*, Miguel León*
*Mälardalen University,Västerås, Sweden,
{bjorn.leander, tijana.markovic, miguel.leonortiz }@mdu.se
†ABB AB, Process Control Platform,Västerås, Sweden

***Abstract*—Modular automation provides a challenge for traditional physics simulators, especially if they are used as a simulator in the loop of a development or research project looking at behavior from a systems level. In this paper, we present extensions of a previously developed simulation environment that is tailored to provide these characteristics. The extensions include simulation engine level improvements, such as including better modeling of the material flow, and sensor anomaly injections to model sensor faults or tampering, as well as system-level enhancements and functionality including certificate handling and anomaly detection methods using machine learning. This simulation environment has proven useful for education as well as research and engineering work, and with the provided extensions several new directions of use can be envisioned. The system is demonstrated in the use case of a modular ice-cream factory, including all the new and enhanced functionalities.**

I. Introduction

To enhance flexibility and dynamicity, supporting mass customization, shortened innovation cycles and novel business requirements, manufacturing systems are being developed in new directions [1]. One of the design strategies being adopted in the process industry domain is Modular Automation (MA) [2], [3]. Systems designed according to MA use individual semi-autonomous modules, which are interconnected through standardized physical and logical interfaces. The low-level control of the process is up to the individual modules, on the other hand, the high-level synchronization of the process is controlled by orchestration units, using formalized workflows or recipes. In this way, the MA system can be scaled and adapted to suit fluctuating operational requirements.

When conducting research and developing products for industrial systems, a common practice is to use a simulator in the loop, providing the behavior of a physical system. However, some characteristics of a MA system are difficult to catch in a traditional physics simulator such as Modelica or MATLAB. Specifically, in MA a system of controllers is connected to the same physical process, individually controlling separate interconnected modules of that process. No available simulators, to the best of our knowledge, can be used as a physical counterpart for several controllers while maintaining the material flow between modules.

Our previous work [4] introduces a simulation environment which sufficiently faithfully simulates a MA system. In the simulation engine, physical modules and module interconnections are defined in an XML file, along with sensors, actuators, and sensor characteristics with the goal to allow easy reconfiguration of the simulator. In runtime, the module behavior is executed, and the sensor and actuator signals are communicated using the Message Queuing Telemetry Transport (MQTT) [5] protocol. The simulator is put in a system environment, including controllers, orchestration units, and supervision, which communicates using the Open Process Communication Unified Automation (OPC UA) [6] protocol, that is being increasingly adopted in industrial control system components. Demonstration of the simulation environment is done in a modular ice-cream factory system, designed for research and education on cybersecurity in manufacturing systems.

This work extends the simulation engine and system environment with crucial functions and components, making it more realistic and useful for research purposes and prototype development.

The simulation engine is extended with functionalities related to different internal and external threats that can affect the normal behavior of the system. In particular, a dedicated anomaly injection component is developed to allow users to intentionally introduce anomalies into the signals of various analog sensors during the production process. When discussing anomalies, we are referring to data points or patterns in the data that deviate from the expected or normal behavior. These anomalies can occur due to various factors, including malicious activity or system failures [7]. Additionally, to allow for higher flexibility and scalability with regards to module integration, a complex material flow [8] handling is implemented in the simulation engine. This allows a generic description of material mixing, separated from the behavior modeling of the physical modules. Also, a method for describing downstream flow throttling is integrated into the material flow handling, which was previously missing. In the prior version", there was no way of describing physical limitations of the inlet flow to a module if it was dependent on an output from other modules.

On the other hand, the system functions are also extended with different components. Firstly, when introducing new modules, as well as other digital services, within the simulation environment there are a number of actions required for them to be accepted and recognized in the system. These actions were previously done by manual certificate distribution, which is a complex and error-prone task, especially when the system is growing in complexity. To that avail, we have described

a simplified secure provisioning method and implemented a generic support for OPC UA services for transitioning between provisioning and production mode. Secondly, an orchestration client is included with a user interface that allows the usage of different recipes as well as the supervision of the current status. Lastly, an anomaly detection component is developed with the goal to generate an alert when an anomaly occurs within the environment. We decided to employ Machine Learning (ML) algorithms for performing the detection, given their considerable success in addressing similar problems across various domains in recent years [7], [9], [10].

The main contributions presented in this paper are the following extensions of:

- Simulation Engine
  - Support for complex and throttled material flow.
  - Anomaly injection
- Simulation system
  - A light-weight provisioning protocol, integrated with certificate management.
  - A recipe orchestration client.
  - Anomaly detection

The remainder of this paper is organized as follows. Section II provides details on the implementation of new components of the simulation engine. Section III provides information on new system-level functionalities. Section IV contains the demonstration of the simulation environment on the example of a modular ice-cream factory. In Section V we discuss the potential applications, advantages and limitations, and outline the future plans. Section VI concludes the paper.

II. Simulation engine

In this section, contributions related to simulation engine are further described.

A. Complex Material Flow

Modeling how the physical properties of the material propagate in the simulator is challenging. Previously, the architecture was limited to communicate using scalar values, e.g, volumetric flow (m^3/s) and temperature. For simulation models utilizing just a few physical properties, this was a simple and useful method. However, when new physical properties needed to be added, the amount of scalar signals needed to describe the system increased, and the logic for propagation of properties when mixing materials typically ended up being duplicated in several modules.

To alleviate this a special type of signal is introduced and defined as complex material flow. The complex material flow signal is defined as a hash-table, with pairs of $(name, value)$. Each module does not need to know about all aspects of the complex material flow, properties of relevance for the module are accessed and manipulated using their name.

When material is flowing from one module into another, a mixture of materials is occurring. To allow this mixture without modules needing detailed knowledge of the properties of the complex material flow, a separate mixing engine is introduced. For a specific set of physical properties, this method must be implemented, so that the mixing is sufficiently precise and physical property interactions are properly modeled.

Below, an example of a subset of the calculation for mixing two materials from the ice-cream factory, with V being the volume, ρ the density, T temperature and C the heat capacity:

$$\begin{bmatrix} V_1 \\ \rho_1 \\ T_1 \\ C_1 \end{bmatrix} + \begin{bmatrix} V_2 \\ \rho_2 \\ T_2 \\ C_2 \end{bmatrix} = \begin{bmatrix} \Delta V \\ \Delta \rho \\ \Delta T \\ \Delta C \end{bmatrix}$$

where ΔV, $\Delta \rho$, ΔT, ΔC represents the resulting value of the mixing of material 1 and material 2, with respect to volume, density, temperature, and heat capacity, respectively. Specifically, these values are calculated by the following equations:

$$\begin{aligned} \Delta V &= V_1 + V_2, \\ \Delta \rho &= \frac{V_1\rho_1 + V_2\rho_2}{\Delta V}, \\ \Delta T &= \frac{T_1C_1 + T_2C_2}{C_1 + C_2}, \\ \Delta C &= \frac{C_1V_1\rho_1 + C_2V_2\rho_2}{V_1\rho_1 + V_2\rho_2} \end{aligned}$$

As can be seen, this is a simplified way of describing mixing of materials, as we assume instant homogeneous mixture without phase transfers, etc. For the simple scenario this is sufficiently precise, but for other applications these formulas would need to be elaborated. That is also supported in the configuration of the simulator, with selection of calculation engine being one of the introduced parameters.

B. Flow throttling

Flow of material in to a module has only been limited by the output from the previous interconnected module(s). However, the volume-flow between modules are in reality limited by the physical properties of the interconnecting infrastructure joining the modules, i.e., both the out-flow and in-flow of a module are dependent not only on properties of the module, but also on properties of the connected module. For example, as illustrated in Fig. 1, the outlet of a tank t_1 is connected to another tank t_2 through a pipe with a variable valve control, meaning that the output of t_1 is depending both on gravitational pressure, outlet geometry, and the opening level of the valve. This behavior is now modeled as flow throttling, which simply implies that the output of a module may be limited to a certain, potentially variable, value. Flow throttling is part of the signal configuration of the simulator configuration files.

C. Anomaly Injection

The simulator has a functionality that enables injection of anomalies in the analog sensors, which can occur during the production process itself. Access to this functionality is enabled through the graphical user interface where the user

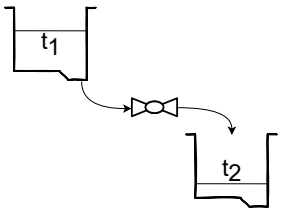

Fig. 1: Flow throttling.

can select type of an anomaly, corresponding parameters for that anomaly, a module, an analog sensor of that module, and a moment when the anomaly will be injected (immediately or after the sensor has reached a specific value either with an increasing or a decreasing trend). Additionally, there is a production process visualization where the user can observe how it is influenced by the anomaly. From the moment of anomaly injection, the actual sensor value is replaced by a modified value by inserting it directly into the MQTT queue. This means that the ”physical” state of the system is not affected, but the controllers access the wrong values, which causes changes in the system behavior. The values can be modified in different ways: freeze value, step change and ramp change [11]. This functionality can be used to simulate different scenarios, from malfunctioning sensors to external intrusions.

III. System functions

In this section, system-level contributions are outlined.

A. Certificates and Provisioning

Secure provisioning is the method of providing a digital entity the knowledge and material required in order to be functional in a specific system. Provisioning mode is when an entity is waiting to be provisioned. In this state the server has no reliable way of knowing how a legitimate client looks like, and will often need to trust any client attempting to connect.

The OPCfoundation .NET stack[1], which most server functions in the system are developed with, contains no built-in implementation of provisioning mode. For the access control mechanisms to work, certain aspects of the provisioning must be in place, especially with regards to certificate handling. Therefore, a light-weight provisioning functionality is provided. The implementation is inspired by the OPC UA Part 21[2], which outlines device-onboarding practices, but our implementation is limited to the functionalities required related to access control.

The goal of the virtual provisioning implementation and accompanying protocol is to introduce an OPC UA server in the system and provide it with any needed information. This is done by replacing the self-signed server application instance certificate with a certificate signed by the Cerificate Authority (CA), as well as by providing additional required public certificates: for the issuing CA, and potentially other CAs, and the token-signing certificate that the authorization service will use for access tokens.

We created an extended base-class of the OPC UA BaseServer, which checks the application instance certificate on startup. If the certificate is self-signed, i.e., the issuer of the certificate is the same as the subject, we consider the server not being provisioned yet, and therefore it allows connections from clients with non-trusted certificates (from the server perspective), following the principle on trust on first use. In this state only the server configuration capabilities are available. The log-in is protected by a username/password.

When the server application instance certificate is not self-signed at startup, the server is considered being in production mode and therefore also start the node manager(s) exposing server-specific functionality. In this mode, the server only accepts connections from trusted certificates, or certificates signed by trusted CAs.

This light-weight provisioning scheme is not intended for actual production environments, as several components are missing, but it can be used as a bare minimum on how to onboard OPC UA servers by means of distributing common root certificates, and providing the server with a CA-signed certificate.

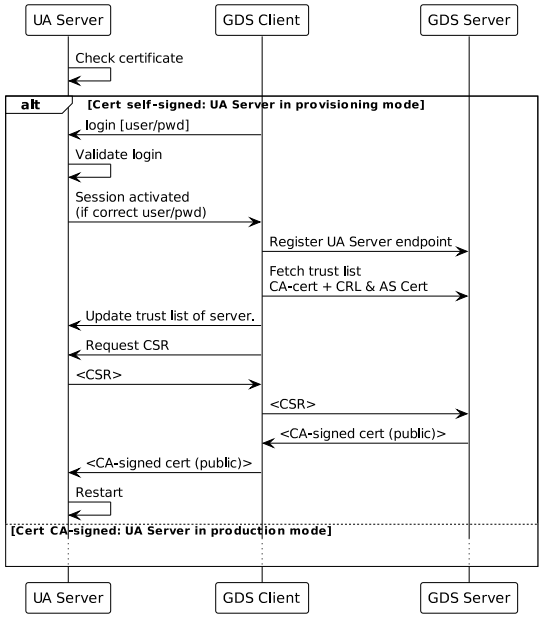

Fig. 2: Provisioning protocol.

For the CA functionality, the built-in CA part of the Global Discovery Server (GDS) example-application provided in the OPC UA .NET-stack[3] is used. The provisioning protocol is visualized in Fig 2. When the UA server starts, it first check the status of its application instance certificate, and either enters provisioning mode or production mode. If in provisioning mode, only the server configuration methods are available. In

[1] github.com/OPCFoundation/

[2] OPC UA v1.05.02 10000-21 Device onboarding (reference.opcfoundation.org/Onboarding/v105/docs/).

[3] github.com/OPCFoundation/UA-.NETStandard-Samples

this state a GDS client can connect to the UA server, register it with the GDS, and update the trust list of the UA Server, i.e., provide it with different root certificates, certificate revocation lists, etc. After that, the GDS Client can order the UA Server to issue a Certificate Signing Request (CSR), which is handled by the CA part of the GDS. When the certificate of the UA Server is updated, the server must restart to transition into production mode.

B. Orchestration client

In order for operator to run recipe execution and monitor the orchestration status, a simple orchestration client is implemented (Fig 3). The orchestration client is part of the operator workplace, and provides a graphical user interface that can interact with the OPC UA server endpoint of the orchestrator units in the system. Using the orchestration client, recipes can be started, stopped, and recipe status can be supervised.

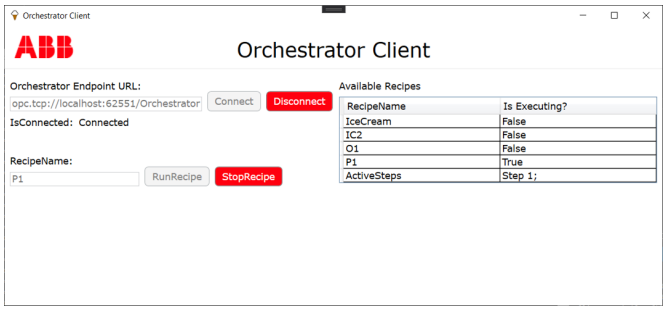

Fig. 3: Graphical user interface for orchestration client.

C. Anomaly Detection

The new functionality is developed to perform detection of anomalies in the production process. This functionality uses artificial intelligence to generate the alerts when an anomaly occurs. More precisely, it is using Long Short-Term Memory (LSTM) artificial neural network to perform binary classification based on the current system state. This algorithm was the most suitable since it considers the temporal nature of the manufacturing process. Additionally, runtime adaptation of LSTM predictions is implemented to prevent false negative and false positive alarms as much as possible [12]. When the alarm happens it remains on until the run ends or until someone checks the systems.

IV. Demonstration

The feasibility of the simulation environment is demonstrated in a use case of a modular ice-cream factory, illustrated in Fig. 4, which is an extension of the use case provided in [4]. The following subsections demonstrate the new functionalities in the scope of ice-cream factory use case.

A. Homogenization using flow throttling and complex material flow

The simulation engine is extended with a separate homogenizer module, which implements the flow throttling, as the homogenization is modeled as the material passing through two valves under high preassure, i.e., there is no internal storage tank part of the homogenization module, it is just material flow. The complex material flow is a generalized functionality, and thus integrated in the simulation engine, with material mixing including heat transfer, fat globule size, sugar content, density and a simplified model for bacterial growth.

B. Provisioning and certificate handling

The GDS represent the basic functions related to the provisioning protocol and certificate handling. All the OPC UA services in the system are provisioned using the protocol. The Orchestrator unit is supervised and controlled by the Orchestrator client, being run as part of the Operator Workplace.

The Authorization Service in the system is used for making access control decisions on behalf of the resource servers in the system, following a policy delegation protocol further described in [13], which is utilizing an approach based on signed JSON Web Tokens (JWT) [14]. The integration of this access control functionality in the demonstration is relying on the certificate distribution part of the provisioning protocol, e.g., related to verifying the JWT signatures.

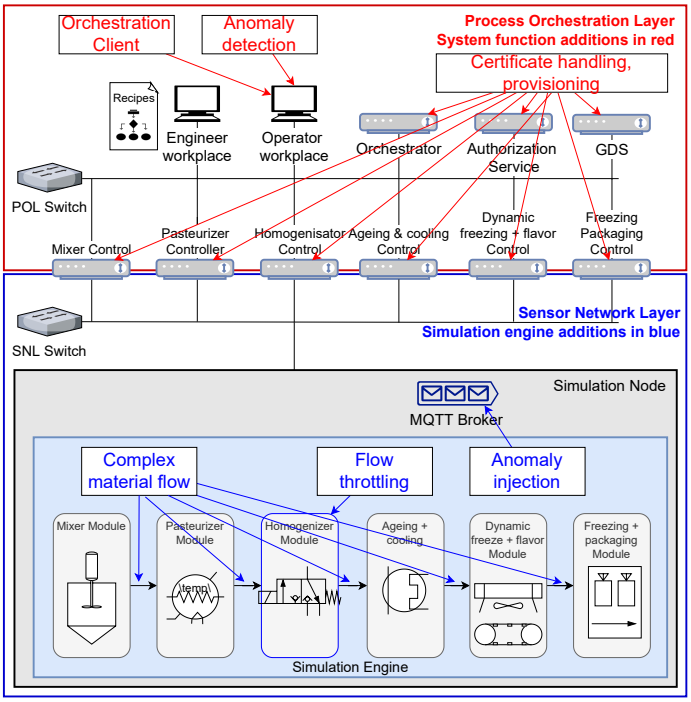

Fig. 4: The modular ice-cream factory, the use case for the simulation environment, with all extensions highlighted.

C. Dataset

The developed simulation environment and the simulation of ice cream-making process with anomaly injection functionality were used to record a dataset named Modular Ice-cream Factory Dataset on Anomalies in Sensors (MIDAS) [11] that is publicly available on GitHub[4]. The main purpose of the

[4]https://github.com/vujicictijana/MIDAS/

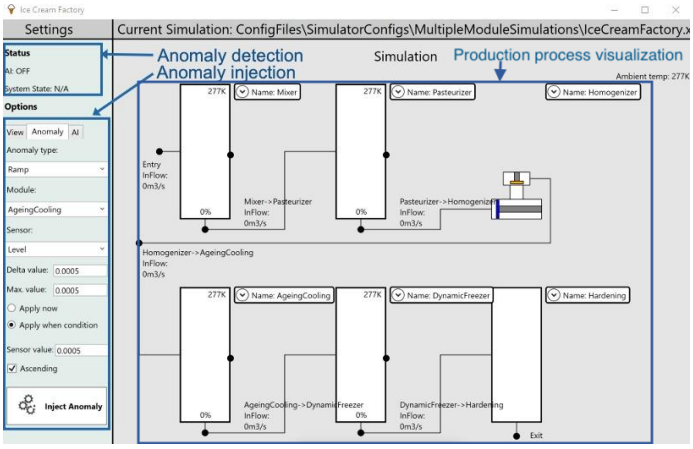

Fig. 5: Example of simulation environment graphical user interface including: production process visualization, anomaly injection and anomaly detection

TABLE I: Accuracy of different ML algorithms anomaly detection and anomaly classification [11], [12]

ML algorithm	Anomaly detection	Anomaly classification
LR	64.60	51.42
DT	69.97	56.53
RF	75.07	62.49
MLP	82.36	70.73
LSTM	**85.06**	**70.85**

dataset is to support machine learning research in analog sensor data for manufacturing environments. The system behavior was recorded during 1000 runs (258 normal runs and 762 anomalous runs), resulting with 36,124,859 instances in the dataset. Anomalous behavior was simulated by injecting different types of anomalies into signals of analog sensors of all modules. Values of different sensors were modified during different stages of the simulated process using different values of parameters for a specific anomaly that was injected.

D. Anomaly detection using machine learning

The MIDAS dataset was used to evaluate different ML algorithms to find the best performing one that can be used for anomaly detection functionality of the simulator. The algorithms that were evaluated include four traditional, non-time-series, ML algorithms (Decision Tree (DT), Random Forest (RF), Multilayer Perceptron (MLP) and Logistic Regression (LR)) and one time-series ML algorithm (LSTM). Two different problems were addressed (anomaly detection and anomaly classification) and the results are presented in Table I.

The results achieved by the ML algorithms showed that MLP is the best non-time-series ML algorithm to detect and classify these anomalies. However, the LSTM has better performance than all non-time-series ML algorithms it was compared with. That is why LSTM was used to develop the anomaly detection functionality presented in III-C. More details on this experiments can be found in [11] and [12].

V. Discussion

This section suggests the potential use of our solution and discusses its advantages and current limitations, as well as planned future extensions.

The developed simulation environment has many advantages as it can be used for research and educational purposes. There are not many tools that can be used to simulate a modular manufacturing system, so this tool and the data set generated with it give the opportunity to the researchers in manufacturing environments to further advance this research area. The simulation environment has already been used for many student projects and master thesis, and it will be used further. Additionally, it can be used in the industrial context, to create a digital twin for a real factory, or as simulator in the loop for development projects within industrial control.

The proposed simulation engine is not meant to be high-fidelity simulator, the goal is to have sufficiently accurate behavior, which is a limitation making the system unfit for, e.g., fine-tuning of control logic etc. The provisioning protocol is a highly simplified variant of what would be required for secure provisioning in a real scenario, but rather serves as the basic functionality needed to support secure communication and access control. When it comes to anomaly injection the current version supports only a limited amount of anomalies that can be simulated, and the anomalies can be injected only in analog sensors. However, the code can be easily extended with the simulation of new types of anomalies, as well as the simulation of the anomalies in the digital sensors. In the same direction, for anomaly detection/classification a limited amount of ML algorithms were tested and only the best one was integrated into the tool. The public dataset can be used to evaluate more ML algorithms and any of them can be integrated into the tool.

The simulation environment can be further extended with additional functionalities. Some extensions that are already planned include:

- **Federated/Distributed learning** [15], [16] - This environment supports a perfect scenario to apply federated or distributed learning, due to the modularity of the system. ML models can be installed into the different modules, to perform a first level of anomaly detection/classification. Then, these individual ML models can be used to create a global ML model on the orchestrator to have a second security level.
- **Feature selection** [17], [18] - The current version of anomaly detection uses values from all the sensors in order to make the prediction. It is planned to evaluate how using only specific sensor values (the most relevant features) affects the performance of ML models. Additionally, feature selection algorithms can help us to determine which sensor are affected by certain anomalies.
- **Industrial controller connectivity** [19] - Currently, the controllers being integrated in the simulation environment

are not real industrial controllers. Work has started to implement MQTT connectivity for one of the control platforms of ABB, which would allow integration of industral controllers for the detailed control of the modules.
- **Workflow-based Authorization** [20] - As MA systems are modular and dynamic to their nature, configuring and sustaining access control policies close to the principle of least privilege is a huge challenge. Tightly integrating knowledge of executing workflows as part of the authorization protocol is currently on-going work, which could be integrated and evaluated in the simulation environment.

VI. Conclusions

In this paper, we present the extensions of the previously developed simulation environment with respect to simulation engine functionalities and main system functions. Extensions of the simulation engine are done with regard to advanced material flow and sensor anomaly injections. Extensions of the system environment are done mainly in the area of cybersecurity, such as secure provisioning and anomaly detection.

The developed environment can be used to simulate the physical process of a modular automation system which is demonstrated by the example of a modular ice-cream factory. The developed solutions benefit the research community, education, and industrial applications. Specifically, the system has shown useful for research on industrial cybersecurity, with several articles and master thesis on intrusion detection and access control for industrial systems being conducted within the InSecTT EU Project[5].

As discussed in the previous section, we are planning to further extend the simulation environment with federated/distributed learning, feature selection, industrial controller connectivity, as well as workflow-based authorization.

Acknowledgements

Filip Johansson, Abhinav Sasikumar, Andreas Ununger and Gloria Ninsiima made technical contributions to the implementation, together with students in the Distributed Software Development course at Mälardalen University, fall semester 2021.

This work is supported by ABB AB; the industrial postgraduate school Automation Region Research Academy (ARRAY), funded by The Knowledge Foundation; and the Horizon 2020 project InSecTT. InSecTT (www.insectt.eu) has received funding from the ECSEL Joint Undertaking (JU) under grant agreement No 876038. The JU receives support from the European Union's Horizon 2020 research and innovation programme and Austria, Sweden, Spain, Italy, France, Portugal, Ireland, Finland, Slovenia, Poland, Netherlands, Turkey.[6]

[5]https://www.insectt.eu/

[6]The document reflects only the author's view and the Commission is not responsible for any use that may be made of the information it contains.

References

[1] A. Sigov, L. Ratkin, L. A. Ivanov, and L. D. Xu, "Emerging enabling technologies for industry 4.0 and beyond," *Information Systems Frontiers*, pp. 1–11, 2022.

[2] NAMUR Working Group 1.12, "NE 148 Automation Requirements relating to Modularisation of Process Plants," NAMUR-recommendation, 2013.

[3] ZVEI—German Electrical and Electronic Manufacturers' Association, "Process industrie 4.0: The age of modular production," White Paper, August 2019.

[4] B. Leander, T. Marković, A. Čaušević, T. Lindström, H. Hansson, and S. Punnekkat, "Simulation environment for modular automation systems," in *IECON 2022–48th Annual Conference of the IEEE Industrial Electronics Society*, pp. 1–6, IEEE, 2022.

[5] "MQTT Version 5.0," OASIS Standard, March 2019. Edited by Andrew Banks, Ed Briggs, Ken Borgendale, and Rahul Gupta.

[6] "IEC 62541 OPC unified architecture, rev 1.05," standard, International Electrotechnical Commission, Geneva, CH.

[7] V. Chandola, A. Banerjee, and V. Kumar, "Anomaly detection: A survey," *ACM computing surveys (CSUR)*, vol. 41, no. 3, pp. 1–58, 2009.

[8] S. Hoher, P. Schindler, S. Göttlich, V. Schleper, and S. Röck, "System dynamic models and real-time simulation of complex material flow systems," in *Enabling Manufacturing Competitiveness and Economic Sustainability: Proceedings of the 4th International Conference on Changeable, Agile, Reconfigurable and Virtual production (CARV2011), Montreal, Canada, 2-5 October 2011*, pp. 316–321, Springer, 2012.

[9] S. Omar, A. Ngadi, and H. H. Jebur, "Machine learning techniques for anomaly detection: an overview," *International Journal of Computer Applications*, vol. 79, no. 2, 2013.

[10] A. Patcha and J.-M. Park, "An overview of anomaly detection techniques: Existing solutions and latest technological trends," *Computer networks*, vol. 51, no. 12, pp. 3448–3470, 2007.

[11] T. Markovic, M. Leon, B. Leander, and S. Punnekkat, "A Modular Ice Cream Factory Dataset on Anomalies in Sensors to Support Machine Learning Research in Manufacturing Systems," *IEEE Access*, vol. 11, pp. 29744–29758, 2023.

[12] T. Markovic, A. Dehlaghi-Ghadim, M. Leon, and S. Punnekkat, " Time-series Anomaly Detection and Classification with Long Short-Term Memory Network on Industrial Manufacturing Systems ," in *18th Conference on Computer Science and Intelligence Systems FedCSIS*, 2023.

[13] B. Leander, A. Čaušević, T. Lindström, and H. Hansson, "Access control enforcement architectures for dynamic manufacturing systems," in *2023 IEEE 20th International Conference on Software Architecture (ICSA)*, pp. 82–92, 2023.

[14] M. Jones, J. Bradley, and N. Sakimura, "JSON Web Token (JWT)." RFC 7519, May 2015.

[15] T. Markovic, M. Leon, D. Buffoni, and S. Punnekkat, "Random forest based on federated learning for intrusion detection," in *IFIP International Conference on Artificial Intelligence Applications and Innovations*, pp. 132–144, Springer, 2022.

[16] H. B. McMahan, E. Moore, D. Ramage, S. Hampson, and B. A. y Arcas, "Communication-efficient learning of deep networks from decentralized data," in *International Conference on Artificial Intelligence and Statistics*, 2016.

[17] I. Guyon and A. Elisseeff, "An introduction to variable and feature selection," *Journal of machine learning research*, vol. 3, no. Mar, pp. 1157–1182, 2003.

[18] Z. Zhao, F. Morstatter, S. Sharma, S. Alelyani, A. Anand, and H. Liu, "Advancing feature selection research," *ASU feature selection repository*, pp. 1–28, 2010.

[19] S. Opacin, L. Rizvanonic, B. Leander, S. Mubeen, and A. Čaušević, "Developing and Evaluating MQTT Connectivity for an Industrial Controller," in *12th IEEE Mediterranean Embedded Systems Conference (MECO 2023)*, 2023.

[20] B. Leander, A. Čaušević, H. Hansson, and T. Lindström, "Toward an ideal access control strategy for industry 4.0 manufacturing systems," *IEEE Access*, vol. 9, pp. 114037–114050, 2021.