ZEROSWARM

D5.3 - Adaptation of IEC 61499 formalism & compilation time to ease the embedding of AI-focused applications

ZERO-enabling Smart networked control framework for Agile cyber physical production systems of systems

Project funded by Horizon Europe, Grant Agreement #101057083

Topic	HORIZON-CL4-2021-TWIN-TRANSITION-01-08
Project Title	ZERO-enabling Smart networked control framework for Agile cyber physical production systems of systems
Project Number	101057083
Project Acronym	Zero-SWARM
Contractual Delivery Date	M14
Actual Delivery Date	M15
Contributing WP	WP5
Project Start Date	01/06/2022
Project Duration	30 Months
Dissemination Level	Public
Editor	INNO
Contributors	all

Author List

Leading Author (Editor)			
Surname	Initials	Beneficiary Name	Contact email
Pagliarini	GP	RPK	GianPietro.Pagliarini@promachbuilt.com
Co-authors (in alphabetic order)			
Surname	***Initials***	***Beneficiary Name***	***Contact email***
Maccioni	RM	RWG	Raffaele.Maccioni@Rwings.tech
Contributors (in alphabetic order)			
Surname	***Initials***	***Beneficiary Name***	***Contact email***
Fritz	AF	NX-SE	artur.fritz@se.com
Serra	SD	RWG	Domenico.Serra@modelite.tech
Lepore	ML	RWG	Mario.Lepore@modelite.tech
Liakh	TL	LTU	tatiana.liakh@ltu.se
Bastidas-Cruz	AB	FhG	arturo.bastidas-cruz@ipk.fraunhofer.de
Shariati	SB	FhG	behnam.shariati@hhi.fraunhofer.de
Safari	BS	FhG	pooyan.safari@hhi.fraunhofer.de
Del Maso	GD	TTS	dalmaso@ttsnetwork.com

Reviewers List

List of reviewers (in alphabetic order)			
Surname	Initials	Beneficiary Name	Contact email
Domenico	DS	RWG	Domenico.Serra@modelite.tech
Sepulcre	MS	UMH	msepulcre@umh.es
Javier	JM	UMH	j.gozalvez@umh.es
Lucas	CL	UMH	m.lucas@umh.es

Martínez	RM	UMH	rpascual@umh.es
Mpatziakas	AM	CERTH	ampatziakas@iti.gp
Khodashenas	PS	HWE	Pouria.khodashenas@huawei.com

Document History

Document History			
Version	Date	Author	Remarks
0.0	15/02/2023	Pagliarini	Table of Content
0.1 – 0.6	15/09/2023	Pagliarini	Intermediate reviews with contributions by partners
0.7 – 0.8	22/09/2023	Pagliarini	Review for Internal reviewers
1.0	29/09/2023	Pagliarini	Final

DISCLAIMER OF WARRANTIES

This document has been prepared by Zero-SWARM project partners as an account of work carried out within the framework of the contract no 101057083.

Neither Project Coordinator, nor any signatory party of Zero-SWARM Project Consortium Agreement, nor any person acting on behalf of any of them:

- makes any warranty or representation whatsoever, express or implied,
    - with respect to the use of any information, apparatus, method, process, or similar item disclosed in this document, including merchantability and fitness for a particular purpose, or
    - that such use does not infringe on or interfere with privately owned rights, including any party's intellectual property, or
- that this document is suitable to any particular user's circumstance; or
- assumes responsibility for any damages or other liability whatsoever (including any consequential damages, even if Project Coordinator or any representative of a signatory party of the Zero-SWARM Project Consortium Agreement, has been advised of the possibility of such damages) resulting from your selection or use of this document or any information, apparatus, method, process, or similar item disclosed in this document.

Zero-SWARM has received funding from the European Union's Horizon Europe research and innovation programme under grant agreement No 101057083. The content of this deliverable does not reflect the official opinion of the European Union. Responsibility for the information and views expressed in the deliverable lies entirely with the author(s).

Executive Summary

Deliverable 5.3 aims to seamlessly integrate AI features into the IEC 61499 framework, enhancing the capabilities of industrial automation systems. This integration encompasses a wide range of industrial functions, including task scheduling, predictive maintenance, quality assurance, and energy efficiency. The synergy between AI and IEC 61499 has the potential to transform traditional industrial practices. These advanced features exemplify the harmonious relationship between AI and industrial standards, paving the way for more agile, efficient, and intelligent industrial processes.

This deliverable presents a comprehensive strategy for embedding AI functions and offers best practices for developing Function Blocks (FB) within the IEC 61499 development environment. It covers the integration of AI platforms with IEC 61499 Edge Gateways and outlines the workflow of AI in conjunction with the IEC 61499 Engineering Tool. Additionally, it addresses critical cybersecurity considerations. These integrations represent a novel contribution of this deliverable, providing a roadmap for the seamless coexistence of AI and IEC 61499.

IEC 61499, as a standard for distributed industrial processes, measurement, and automation, provides a framework for designing and operating systems using the Function Block (FB) approach. The core innovation element of this deliverable showcases Control Platform compliance with IEC 61499 and related functionalities. This is demonstrated through an industrial case study in the food packaging sector, along with scenarios involving the scheduling and rescheduling of Autonomous Mobile Robots (AMR), which serve as validation scenarios for advanced AI algorithms and control architectures from D2.2.

Table of Contents

List of Figures

List of Acronyms

Acronym	Description
5G	Fifth-Generation Wireless Communications
AE	Alarm And Events
AFoF	AALTO Factory of the Future
AGV	Autonomous Guided Vehicles
AI	Artificial Intelligence
AIC	Automotive Intelligence Centre
AIIC	AALTO Industrial Internet Campus
AR	Augmented Reality
CAT	Composite Automation Type
CPSoS	Cyber-Physical System of Systems
CUC	Centralized User Configuration
DA	Data Access
DFA	Demonstration Factory Aachen (DFA)
DLFi	Distributed Learning Framework
DSS	Dss Dynamic Spectrum Allocation
E2E	End-To-End
eMBB	Enhanced Mobile Broadband
gPTP	Generalized Precision Time Protocol
HDA	Historical Data Access
ICT	Information Communication Technologies
IDS	International Data Spaces
IICF	Industrial Internet Connectivity Framework
IIoT	Industrial Internet Of Things
IIRA	Industrial Internet Reference Architecture
IMC	Intelligent Mechatronic Components
ITU	International Telecommunication Union
LBO	And Local Breakout
MAS	Multi-Agent Systems
MES	Manufacturing Execution Systems
mIoT	Massive Internet of Things
mMTC	Massive Machine Type Communication
MR	Mixed Reality
NASA	National Aeronautics and Space Administration NASA
NASA	Telematic Data Collector
NPN	Non-Public Networks
OLE	Object Linking and Embedding
OPC	Open Platform Communication
OPC-UA	Opc Unified Architecture

PLC	Programmable Logic Controllers
PTZ	Production Technology Center (PTZ)
ROS	Robot Operating System
SLAM	Simultaneous Localization and Mapping
SNPNs	Stand-Alone Npns
SOAP	Simple Object Access Protocol
SoS	Systems Of Systems
TSN	Time Sensitive Networking
UA	Unified Architecture
UACP	Unified Architecture Connection Protocol
UES	User Equipment
UPF	User Plane Function
uRLLC	Ultra-Reliable Low Latency Communication
VR	Virtual Reality
AMR	Autonomous Mobile Robots
ANN	Artificial Neural Network
OR	Operation Research

1 Introduction

IEC 61499 is an international standard for the design and implementation of distributed control systems. It provides a framework for modeling automation systems using a function block approach, promoting modularity and reusability. The standard emphasizes event-driven communication between function blocks, enhancing flexibility and scalability. IEC 61499 aims to improve interoperability and efficiency in industrial automation by offering a standardized approach to system design and integration.

1.1 Purpose of the document

The purpose of Deliverable 5.3 (Adaption of the IEC 61499 Formalism and Compile Time To Easy Embedding of AI) is to integrate AI functionalities within the IEC 61499 framework and significantly enhance the capabilities of industrial automation systems. From job scheduling to predictive maintenance, quality assurance, and energy efficiency, the convergence of AI with IEC 61499 demonstrates the potential to revolutionize traditional industrial practices. These functionalities exemplify the symbiosis between AI and industrial standards, opening the way for more responsive, efficient, and intelligent industrial processes.

IEC-61499 is integral to Zero-SWARM, enabling the seamless integration of AI capabilities into industrial automation systems. It structures AI functionality within function blocks, guides FB development, defines communication in the Edge Gateway Architecture, and facilitates TensorFlow and ONNX integration, ensuring efficient, standardized AI implementation.

There will be described the steps to embed AI functionalities and the best practice for developing FB - Function Block - in IEC 61499 development environment, the integration of AI platform and IEC 61499 Edge Gateway, the AI workflow with IEC 61499 Engineering Tool, including cybersecurity aspects. Such integrations represent the novel contribution of the deliverable.

Note: this deliverable was delayed around 1 month compared to the original delivery due time, because of additional synchronization needs with WP4 and other WP5 tasks.

1.2 Structure of the document

The document is structured as follows:

- **Chapter 1** Introduction:
    - Introduction and structure of the document**;**
- **Chapter 2** Adaption of the IEC 61499 Formalism and Compile Time To Easy Embedding AI functionalities:
    - IEC 61499, a standard for industrial automation, intersects with AI in this document. By embedding AI techniques within the framework, it enhances interoperability and decision-making in distributed systems, ushering in a new era of intelligent industrial automation.
- **Chapter 3** Examples of AI based Functionalities:
    - In this chapter, we explore the integration of AI within the IEC 61499 standard, with a spotlight on its practical application in a production facility featuring AMRs. The goal is to showcase how AI can enhance industrial automation through optimized job

scheduling, predictive maintenance, quality assurance via image recognition, and energy efficiency.

- **Chapter 4** Functionalities of the AI applications:
  - This chapter explores the integration of AI within the IEC 61499 framework. It delves into the role of AI Runtime, which enables the execution of ML models alongside control applications on the Shop Floor, leveraging the Embedded Edge Connectivity Layer (EECL).
- **Chapter 5** Conclusion

2 Adaption of the IEC 61499 Formalism and Compile Time To Easy Embedding AI functionalities

2.1 The steps to embed AI functionalities

IEC 61499 is well-suited to support the realization of the industry 4.0 vision by facilitating the integration of smart manufacturing processes and the development of flexible and adaptive automation systems (Thramboulidis, 2015).

IEC 61499 is a standard for distributed control systems and industrial automation. It provides a framework for designing and implementing control applications using function blocks. These function blocks are versatile building blocks that can be distributed across different devices and serve as the basic units for creating control systems. These modular function blocks can be distributed across different devices, making them suitable for various industrial applications. The increasing adoption of artificial intelligence, including machine learning, and in general of any type of algorithms, such as mathematical optimization algorithms, is creating a significant opportunity for automation systems to improve their efficiency and extend their capabilities distributing intelligent algorithms. The extension of the IEC 61499 to naturally embed such functionalities is a natural step.

Adapting the IEC 61499 formalism for easy embedding of AI: By aligning AI methods with the IEC 61499 standard, it is possible to leverage the interoperability, reconfigurability, and reusability features of the IEC 61499 standard while incorporating intelligent decision-making into distributed industrial automation systems. This approach allows AI models to be encapsulated within function blocks, making it easier to deploy AI in various devices or platforms. Compile-time strategies can be used to optimise the performance and resource allocation of the AI components, ensuring that the embedded AI functions operate efficiently within the constraints of the targeted hardware.

To extend or adapt IEC 61499 to embed AI components or model-based intelligent functions, such as a mathematical optimizer, is possible by implementing the following steps:

1. AI Function Blocks: Define new function blocks specifically for AI-related tasks. These function blocks should encapsulate AI algorithms or any intelligent functions. For instance, function blocks for machine learning models to predict maintenance or mathematical optimizers to schedule jobs (L. Wang, 2006).
2. Modelling AI Components: Determine the input-output interfaces of AI function blocks and specify their behaviour. The behaviour of AI components can be described through state machines, mathematical equations, or other appropriate representations.
3. Integration with Existing Function Blocks: the AI function blocks can interact seamlessly with other standard IEC 61499 function blocks. This involves defining the data exchange

mechanisms and communication protocols between AI components and other control function blocks (S. Campanelli).

4. Handling Data: AI components typically require large amounts of data for training and decision-making. How data can be collected, pre-processed, and fed into the AI function blocks is core. AI models can also output data to control the process or trigger other actions in the control system. It is important to carefully consider data modelling, to make sure that the single intelligent Function Block receives data in the most efficient and "ready-to-use" way. This step might require data pre-elaboration, and data handling within the Zero-SWARM cloud-to-edge platform described in other WPs.
5. Safety and Reliability: When incorporating AI components into industrial systems, safety and reliability are paramount. The implications of AI decisions on the overall system and implementing safety mechanisms and fail-safes to prevent undesirable outcomes are also extremely important, especially in the case of non-explainable ML.
6. Distributed AI: In distributed control systems, AI components might be located on different devices. In such a case, mechanisms are needed to enable communication and coordination among distributed AI function blocks to achieve system-wide optimization or intelligent control.
7. Training and Adaptation: For machine learning-based AI components, it should be considered mechanisms for model training, retraining, and adaptation. It is crucial how the models can learn from real-time data or historical information to improve their performance over time.

2.2 Best Practices for Developing Function Blocks in IEC 61499 with AI Functionalities

Given the significance of formal methods, interoperability, modularity, and reusability as critical aspects (Vyatkin, 2013), here we derive some good practices to follow in the context of Zero-SWARM. Developers should consider successfully integrating AI functions blocks, for example, in Python, managing large data volumes, leveraging the Cyber-Physical System of Systems paradigm, and integrating with broader application layers such as Kafka, MQTT, and OPC-UA within the IEC 61499 framework (Zoitl, 2010).

The recommendations aim to lead to efficiency and robustness in modern industrial automation systems. The integration of AI functionalities involves different aspects:

1. Languages and AI platforms: specifically related to the development languages and platforms used to implement the algorithms.
2. The volume of data: enabling AI functionalities often involves handling a significant amount and heterogeneous type of data.
3. The paradigm of Cyber Physical Systems of Systems: considering that the standard provides a sound basis for the implementation of Cyber-Physical Systems, as required by the fourth industrial revolution (Thramboulidis, 2015) (Liu, 2019).
4. Co-operatibility in cloud-to-edge architectures: modern cloud-to-edge architecture for distributed intelligent control functions, as described in other WPs, requires an application architecture with harmonized different technology layers.

Embedding Functions developed in specific vertical languages such as for example Python:

- Select Suitable Libraries and Environments: Use established libraries and development

environments that facilitate the integration (for example of Python) within IEC 61499.

- Ensure Compatibility: Verify that the chosen Python functions are compatible with the underlying hardware and other integrated technologies.
- Maintain Modularity: Encapsulate the Python functions within distinct Function Blocks to ensure maintainability and reusability.
- Secure the Code: Implement security measures to protect the integrity and confidentiality of the embedded Python code.

Considering the Issues of Significant Amount of Data:

- Optimise Data Handling: Utilize efficient data structures and algorithms to handle large datasets without overloading the system.
- Implement Data Preprocessing: Apply preprocessing techniques to reduce data dimensionality and facilitate faster computation.
- Enable Real-time Processing: If real-time processing is required, ensure that the system is capable of handling large data streams without latency issues.
- Consider Data Storage and Management: Implement robust data management strategies to store and retrieve large datasets efficiently.

Considering the CPSoS paradigm:

- Design Interconnected Systems: Embrace the principles of modularity and interoperability to design systems that can seamlessly interact within a Cyber-Physical System of Systems framework.
- Leverage Distributed Computing: Utilize distributed computing techniques to harness the collective computational power of interconnected systems.
- Prioritize Security and Reliability: Implement robust security measures and ensure the reliability of the communication between different systems.
- Encourage Collaborative Intelligence: Promote information sharing and collaborative decision-making among different subsystems.

Considering the co-operability in cloud-to-edge architectures (interoperability in a Wider Application Layers which might include Kafka, MQTT, and OPC-UA) (Aloqaily, 2019):

- Utilize Standard Protocols: Use standard protocols like OPC-UA to ensure seamless communication between different components, including Kafka and MQTT.
- Ensure Scalability: Design the architecture to support scalability, considering the interaction between IEC 61499 and other application layers.
- Implement Efficient Message Handling: Utilize messaging systems like Kafka and MQTT to handle real-time data streams and ensure efficient message handling.
- Promote Interoperability: Adopt a flexible and modular design approach that enables interoperability between IEC 61499 and other application layers.
- Monitor and Troubleshoot: Implement monitoring and troubleshooting capabilities to quickly identify and resolve issues in the integrated system.

3 Examples of AI based Functionalities

This chapter provides insights into the practical functionalities enabled by the integration of AI within the IEC 61499 standard (Lewis, 2016). A special focus is laid on the use-case of production facilities with an annexed warehouse, where a fleet of AMRs is used.

The aim is to show how AI can be harnessed to enhance various aspects of industrial automation.

Jobs Scheduling with Optimizers:

- Function Block: Scheduler Optimizer
- Functionality: Responsible for optimizing the assignment of jobs to resources such as work centers or robots. AI/ML helps to make situational-aware optimization decisions considering several factors such as energy consumption, availability of resources, etc.
- Integration with IEC 61499: This function block communicates with other IEC 61499 blocks, such as control blocks for low-level functionalities.

Predictive Maintenance:

- Function Block: Maintenance Predictor
- Functionality: Utilizes machine learning algorithms to predict the likelihood of machinery failure and suggests maintenance schedules.
- Integration with IEC 61499: Integrates with monitoring and control function blocks to provide timely warnings and optimize maintenance planning.

Quality Assurance through Image Recognition:

- Function Block: Quality Inspector
- Functionality: Employs computer vision and deep learning to inspect products, detect defects, and ensure quality control.
- Integration with IEC 61499: Communicates with manufacturing process control blocks to provide real-time feedback and corrections.

Energy Efficiency Optimization:

- AI Function Block: Energy Optimizer
- Functionality: Applies optimization techniques to analyse energy consumption patterns and optimize energy utilization across various industrial processes.
- Integration with IEC 61499: Interfaces with different energy consumption points within the industrial facility for comprehensive energy management.

Let's now consider specifically the use-case that is a production facility with an annexed warehouse where an AMR fleet is responsible for autonomously transporting goods from/to production lines and the stock. The goal is to optimize the scheduling and routing of AMRs to maximize efficiency while considering their battery levels to plan recharging effectively.

Jobs Scheduling and Routing with Optimizers: Function Block: Scheduler Optimizer

- Functionality: This function block is responsible for optimizing the assignment of tasks to different AMRs and planning their optimal routes.
- It receives inputs such as the current locations of AMRs, the locations of goods to be transported, and their priorities or deadlines.
- The optimizer employs optimization algorithms (e.g., Genetic Algorithms, Particle Swarm Optimization, etc.) to find the best task assignment and routes for the AMRs in near-real-time.
- The outputs of this block include the optimized schedule and routes for each AMR.
- Integration with IEC 61499: The Scheduler Optimizer function block communicates with the Jobs generator, providing the list of jobs, and with other IEC 61499 function blocks that manage the AMR fleet, such as the AMR Control function blocks responsible for low-level motion control and providing the real-time status.

Prediction of Battery Levels with Machine Learning Model: Function Block: Battery Level Predictor

- Functionalities: this function block is responsible for predicting the battery levels of individual AMRs using a machine learning model.
- It takes inputs such as the current battery level, AMR speed, load, distance to be travelled, and other relevant factors.
- The function block hosts the machine learning model, which was previously trained on historical data to predict the future battery levels (training could be centralised or follow a concept of “federated learning” as described in other WP4 tasks).
- The output of this block is the predicted battery level for each AMR.
- Integration with IEC 61499: It Is deployed on the edge, directly on each AMR's control system. It communicates with the Scheduler Optimizer function block for the cyclical process of jobs assignment which includes recharging planning ensuring that operations continue with minimal disruptions and lifespan of the battery is guaranteed. In addition, the Function Blocks interact with other components and function blocks such as the layout digital twin that is a component providing data and real time information on the layout.

Optimised Job Scheduling and Recharging:

- Functionalities: this function block is an extension of the Jobs Scheduling and Routing with Optimizers function block considering the optimization of recharging.
- The AMRs start their tasks based on the initial schedule provided by the Scheduler Optimizer function block. As the AMRs perform their tasks and move through the facility, they continuously update their positions and status. The Scheduler Optimizer function block receives these real-time updates and uses them to re-optimize the remaining tasks, considering the current positions of AMRs, the predicted battery levels, and any new job requests that come in.
- The Battery Level Predictor function block continuously monitors the AMRs' energy usage and predicts their future battery levels.
- When the Scheduler Optimizer function block detects that an AMR's battery is running low or will be depleted soon, it schedules the AMR for a recharge before it reaches a critical state and considering the operations workload. AMRs are directed to designate charging stations based on their positions and predicted battery levels, ensuring that the operations continue with minimal disruptions.
- Integration with IEC 61499: In addition to the communication described above, for the Jobs Scheduling and Routing function block, this version of the Scheduler also interacts with the Prediction of Battery Levels function block.

4 Functionalities of the AI applications

4.1 Introduction

In the context of Zero-SWARM, the goal is to simplify the engineering process between AI modelers and automation engineers, as well as streamline the creation of IEC 61499 function blocks interacting with the AI system. This optimization step benefits automation engineers for the predictive maintenance use case. The aim is to provide a user-friendly engineering interface for system integrators and facilitate the integration of AI functionality into IEC 61499 function blocks. Partners, such as model creation specialists and vendors of standard modelling software, can contribute to this

objective.

The primary objective in Zero-SWARM is to leverage the IEC 61499 Function Block (FB) concept to integrate various AI models, by utilizing the IEC 61499 Build time (Integrated Development Environment - IDE) for automatic FB-template generation. Additional objective is to establish automatic communication between assets of the IEC 61499 Runtime (RT) and a selected AI Runtime (AI RT) model using OPC UA communication protocols.

In Section 4.2, we pivot towards the AI workflow within the IEC 61499 Engineering tool, offering a comprehensive overview of the use case involving AMRs and their predictive battery management. Subsequently, we delve into the conversion process from TensorFlow to ONNX, elucidating the steps and benefits of this transition.

4.2 AI platform Integration in IEC 61499 Edge Gateway

AI Runtime is a ML execution environment that enables the usage of ML models next to control application on the Shop Floor while allowing connection via various Communication Protocols via integration of EECL (Embedded Edge Connectivity Layer) (Cândido, 2013). Figure 1 explores the IEC 61499 Edge Gateway Architecture, where a central element known as the Intelligent Control Expert block takes center stage.

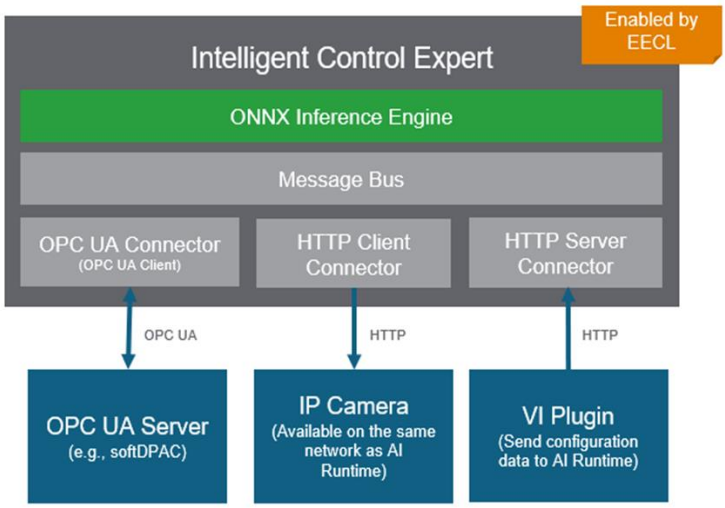

Figure 1 IEC 61499 Edge Gateway Architecture

This block encompasses essential components including the ONNX Interface Engine, OPC UA Connector, HTTP Client Connector, and HTTP Server Connector, each playing a crucial role in enabling advanced AI functionalities. Notably, the Intelligent Control Expert block establishes external connections with key entities such as the OPC UA Server, IP Camera, and VI Plugin, forming a comprehensive ecosystem for seamless AI integration and control.

AI Runtimes can be installed on different platforms depending on your needs and environment, such as personal computers, local servers, public cloud, built-in devices, containers, etc.

In the context of AI, a runtime environment typically includes the necessary software and hardware components to support the execution of AI models or algorithms. These components may include:

1. Framework Support: The runtime environment usually supports one or more AI frameworks (e.g., TensorFlow, PyTorch, scikit-learn) that enable the execution of AI models and algorithms.
2. Hardware Acceleration: To improve performance, AI runtimes may leverage specialized

hardware like GPUs (Graphics Processing Units) or TPUs (Tensor Processing Units) for parallel processing, as AI workloads are often computationally intensive.

3. Inference and Training: The runtime may be optimized for either inference (using a trained AI model to make predictions) or training (updating the model's parameters with new data).
4. Model Format Support: AI runtimes often support common model formats such as ONNX (Open Neural Network Exchange) or TensorFlow Saved Model, allowing users to deploy models generated by different frameworks.
5. Optimizations: Runtime environments may incorporate various optimizations to enhance AI model execution efficiency, such as model quantization, pruning, or caching.
6. APIs and Interfaces: AI runtimes may provide APIs and interfaces that allow developers to interact with the runtime, load models, perform inference, and manage resources.

Figure 2 describes the communication layer between IEC 61499 Edge Gateway & Edge Devices.

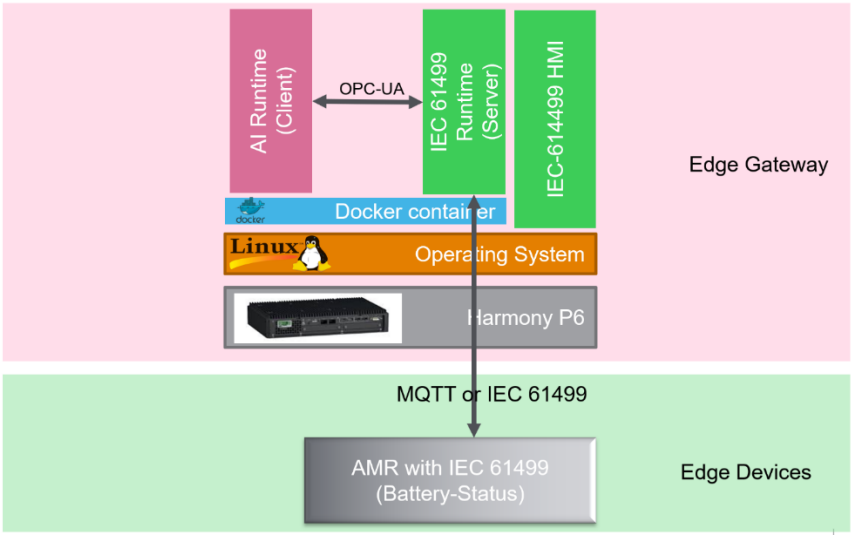

Figure 2 Edge communication layers

For the SN-1,2 Zero-SWARM the Harmony P6 IPC from Schneider electric is used by installing Docker containers to create independent execution environments for the AI models and deploy them in different environments.

4.2.1 What is Tensorflow & ONNX

The discussion of TensorFlow and ONNX in this section serves as a foundational bridge to understanding the integration of AI platforms within the IEC 61499 Edge Gateway architecture. TensorFlow and ONNX play a pivotal role in enabling the execution of machine learning (ML) models alongside control applications on the shop floor. Before delving into the integration process, let's briefly explore what TensorFlow and ONNX are and why they are instrumental in the AI landscape.

ONNX, or Open Neural Network Exchange, is an open-source standard for representing deep learning models. It was developed by Facebook and Microsoft to make it easier for researchers and engineers to move models between different deep-learning frameworks and hardware platforms. One of the main advantages of ONNX is that it allows models to be easily exported from one framework, such as PyTorch, and imported into another framework, such as TensorFlow.

This can be especially useful for researchers who want to try out different frameworks for training and deploying their models, or for engineers who need to deploy models on different hardware platforms. **Error! Reference source not found.** describes the technologies with which ONNX is compatible.

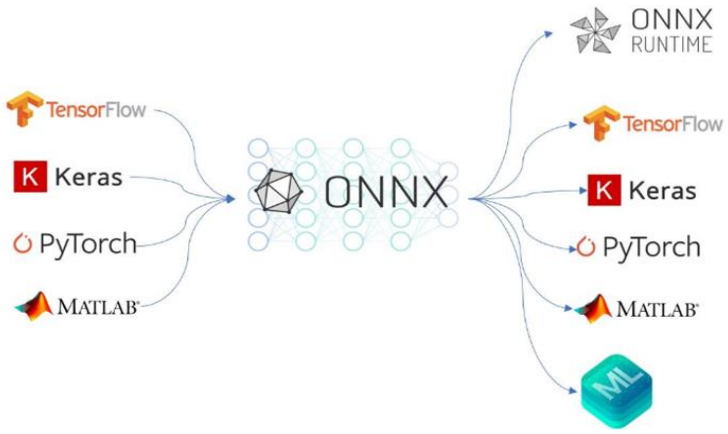

Figure 3 ONNX Technologies

4.2.2 How to convert model from Tensorflow to ONNX

Converting a model from TensorFlow to ONNX (Open Neural Network Exchange) format allows us to use the model with various deep learning frameworks that support ONNX. This can be useful for deploying your model on different platforms or frameworks. Here is how they can convert a model from TensorFlow to ONNX:

1. **Install Required Libraries:**

First, it is needed to make sure to have the necessary libraries installed. Both, TensorFlow and ONNX packages will be needed. It can be installed using pip:

```
pip install tensorflow onnx
```

2. **Export TensorFlow Model:**

Before converting the model, it is needed to export it in a format that can be converted to ONNX. TensorFlow provides the Saved Model format for this purpose. The following figure shows how to save a TensorFlow model:

```
import tensorflow as tf

# Load or create your TensorFlow model
model = ...

# Save the model in SavedModel format
tf.saved_model.save(model, "path/to/saved/model")
```

3. **Convert to ONNX:**

Once the Model is saved, the tf2onnx library is used to perform the conversion. It is needed to specify the input and output nodes of the TensorFlow model that are going to be convert.

```python
import tf2onnx

# Load the TensorFlow model
tf_model = tf.saved_model.load("path/to/saved/model")

# Convert TensorFlow model to ONNX
onnx_model, _ = tf2onnx.convert.from_saved_model(
    "path/to/saved/model",
    input_signature=tf_model.signatures['serving_default'],
    opset=13  # Use the appropriate ONNX opset version
)

# Save the ONNX model to a file
with open("path/to/output/model.onnx", "wb") as f:
    f.write(onnx_model.SerializeToString())
```

2. **Specify Opset Version:**

In the conversion step, you need to specify the ONNX opset version (opset parameter). The opset version determines which ONNX operations are used to represent TensorFlow operations. You can find the compatible opset version by checking the documentation or the ONNX GitHub repository.

3. **Verify & Test:**

After conversion, it's good practice to load the ONNX model and test it using a library that supports ONNX. This will ensure that the conversion was successful, and the model behaves as expected.

4.3 AI workflow with IEC 61499 Engineering tool

Overview of the Use Case

AMRs are versatile robotic systems that play a crucial role in various industries. These robots are capable of autonomous movement and can perform a wide range of tasks, making them suitable for applications in logistics, manufacturing, and beyond. For the safe and uninterrupted operation of an AMR it is necessary to monitor the batteries used in each AMR. Especially in case of high workload, smart planning of recharging is required. In Zero-SWARM the primary focus of the use case is on the utilization of AMRs in Reepack's food packaging production line.

Analysing historical data on battery usage and discharging patterns of AMRs in a factory can lead to the development of predictive models that offer valuable insights into battery management and operational efficiency. This process involves extracting meaningful patterns and trends from past AMR behaviour to anticipate when their battery levels might fall below a predefined threshold. Using machine learning techniques, AI engineers develop predictive models. These models learn from historical data to make accurate predictions about future battery levels. Using these predictive models, the operator or the AMR maintenance person can plan the recharging process in advance to ensure the AMR is recharged in the low activity duration and the production process remains undisturbed.

Figure 4 below shows the visualized concept of the SN-1 trial use case of Reepack described in D5.1, where the "AMR navi" module and IEC 61499 node are communicating with the AI via OPCUA configuration. The data from AMR will be communicated to the AI runtime using IEC 61499 Runtime and vice versa.

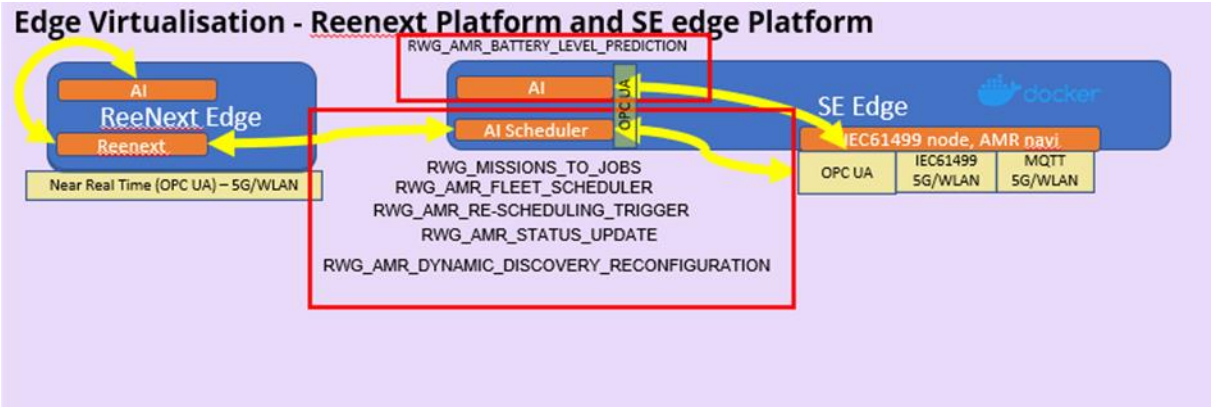

Figure 4 Concept of Reepack Use Case

Within the IEC 61499 platform, the goal is to get the battery status from an AMR such as charging, discharging, state of charge (SOC), etc. as input and provide predictive monitoring to the operator to take further control actions. The inputs from the battery point can be provided to the IEC 61499 platform by creating a Predictive Model, e.g. by Python function, which will be used through the IEC 61499 Runtime (Cengic, 2018).

The Engineering-Workflow for the Automation-Engineer, to connect AI with IEC6 1499 Function Blocks

The provided steps in Figure 5 outline the process of integrating a set of AI functions into the IEC 61499 environment, including importing an AI model or library, creating Function Blocks (FBs) for the control functions, using communication protocols (OPC UA and MQTT) (Mahmoud, 2015) to interact with the AI model/runtime by extending the IEC 61499 library with this functionality.

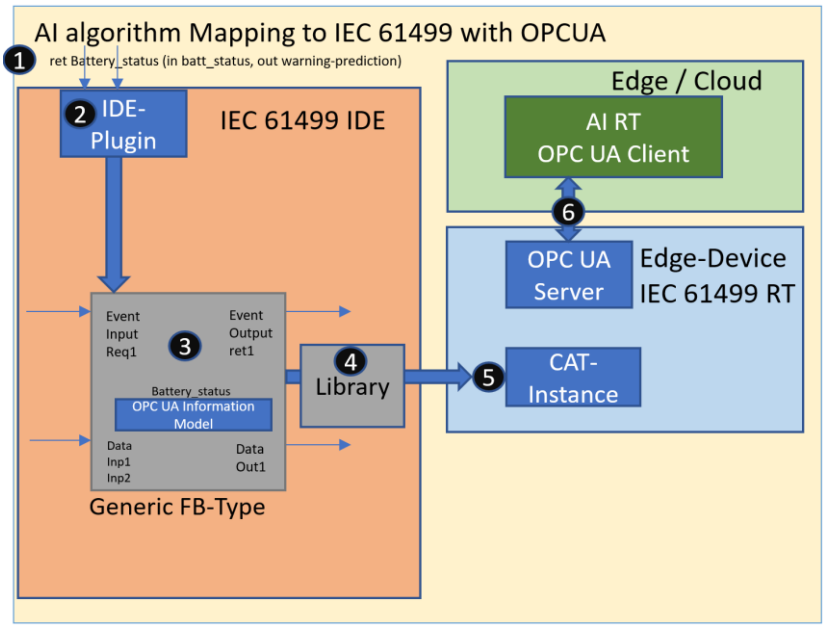

Figure 5 AI Algorithm mapping to IEC 61499 with OPCUA

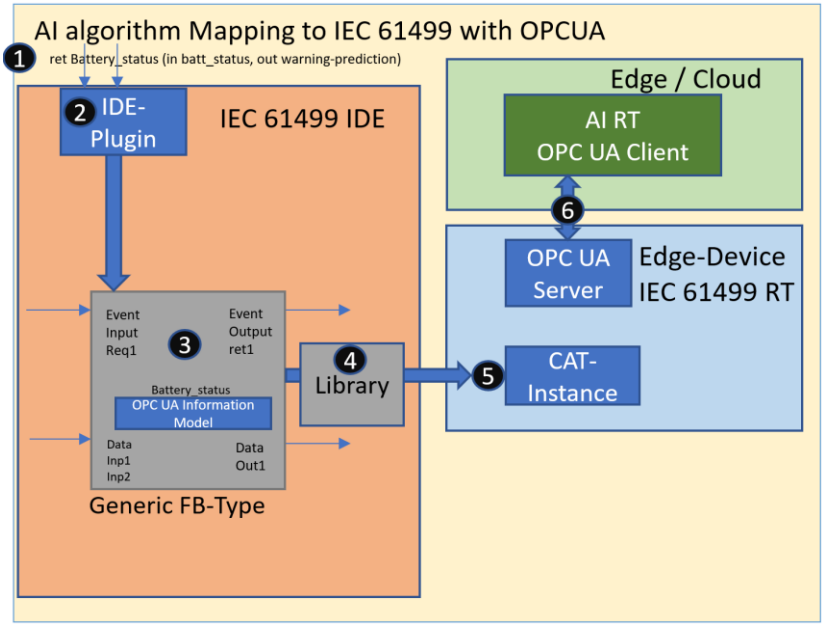

Figure 5 describes how an AI algorithm can be embedded in IEC 61499 using OPC UA. Here is a detailed description of each step:

1. **Set of AI Functions (Library):** This refers to a collection of pre-designed AI functions, made by an *"AI practitioner"*, that perform specific AI tasks, such as data analysis, prediction, or decision-making. These AI functions are usually packaged in a library, then an AI model is created and trained. The ready-to-use AI functions and model are used in step 2.

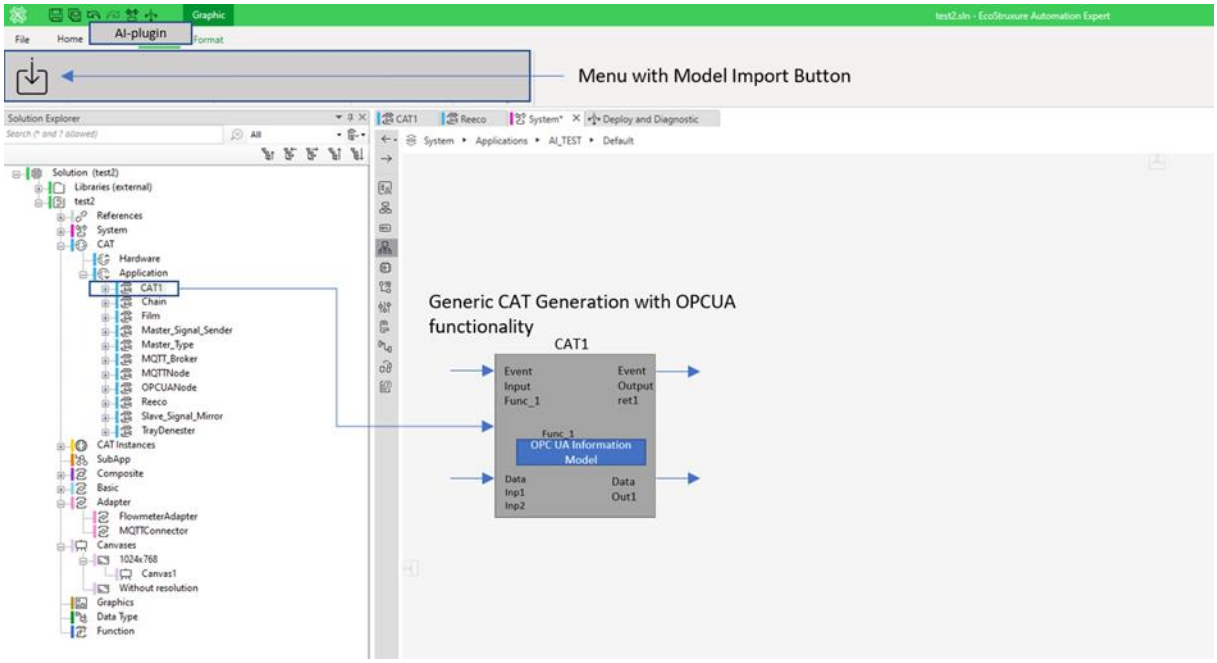

Figure 6 Mockup IDE plug-in implementation in EAE

2. **AI Model Import in IEC 61499 IDE:** In this step, the AI functions/model will be integrated into an IEC 61499 Function Block. An *"advanced Automation-Engineer"* uses the IEC 61499 Integrated Development Environment (IDE). This involves importing the AI function/model, making them visible within the IDE for further selection, able to generate a corresponding function block in step

3. In Figure 6 a mockup of an IDE Import plug-in for the EcoStruxure Automation Expert - Engineering platform (IDE) is shown. Blue highlighted area shows how the plug-in will look when it's integrated into the IDE and used. The plug-in will have a menu with an import button which makes it able to select an AI-Model. During this import AI-functions with input and out parameters are shown, which must be selected/committed by the automation-engineer.

4. **Generation of a generic FB template with OPC UA (MQTT) communication**: After the commitment the AI function with its input and output parameters is converted into a generic and basic FB type with corresponding data-input and data-output parameters, which serves as a template for creating instances of AI Functionality. Additionally, this FB template configures the communication for using IOT protocols like OPC UA or MQTT, allowing communication between AI runtime and the IEC 61499 runtime system for this dedicated AI functionality. For this integration, a Composite Automation Type generation is needed.

5. **Extend an IEC 61499 Library with AI Functions (nested CAT Creation):** In this step, the IEC 61499 library is extended by the *"advanced Automation Engineer"* to incorporate between this AI function and a dedicated technological Function Block, which uses this AI functionality. This is done by creating a nested Composite Automation Type (CAT) within the IEC 61499 library, which is ready to use by a "simple automation Engineer", with lower knowledge about the insights of AI and IEC 61499. Nested CATs are structured entities that define the behaviour and interactions of multiple FB instances.

6. **Instantiating CAT objects within an IEC 61499 Application**: The "normal Automation Engineer" is going to create the IEC 61499 control application, by instantiating those nested CAT objects, without knowing the details of the AI functionality in the background, using the previously created CAT, and dragged into the IEC 61499 application and connecting it over single line engineering with the other technological IEC 61499 objects of the application. These CAT instances represent the integration of AI functions into the IEC 61499 application's logic.

7. **Start IEC 61499 control system and exchange data with the AI Runtime (OPC UA and MQTT):** Upon execution, the IEC 61499 application initiates. Communication between the IEC 61499 application and the AI Runtime (an environment where AI functions are executed) is established using communication protocols such as OPC UA and MQTT. Data is exchanged between the control system and the AI Runtime automatically. This data exchange involves inputs, outputs, and any necessary control signals.

In essence, the above process enables the integration of AI capabilities within the IEC 61499 environment. By generating generic FB types, using IOT communication protocols, extending the library and application, and enabling data exchange, the AI functions become integral parts of the industrial control system, enhancing decision-making and optimization processes. The structured approach ensures that AI functions can be seamlessly integrated and utilized within the existing control logic and automation infrastructure.

4.3.1 Implementation Details for the IEC 61499 IDE-Plugin

The purpose of the new developed IDE plugin is to extend and add new functionality to the IEC 61499

engineering tool by extending the toolbar with buttons like ‘import model’, where an advanced automation engineer will follow the steps from Figure 6.

There are these main points to consider in the implementation of the IDE-Plugin:

- With the model information (Data-input/Output) the IEC 61499 XML-serializer is fed with the AI-function declaration to create the IEC 61499 FB in XML format. In the context of the IEC 61499 environment, XML (Extensible Markup Language) format is utilized to create and define new Function Blocks (FBs). Each time a new FB is created, an XML file gets generated, which contains the basic structure of inputs and outputs of the function intended to call functionality from the AI model. Once the XML format is generated and the FB template is fixed, it is less time-consuming for an advanced automation engineer to further extend or create a nested CAT for the intended application and put the result into an IEC 61499 library.

- Since the ONNX runtime and the IEC 61499 environment might have different data structures and types, the mapping of datatypes between the ONNX AI-Model and the IEC 61499 FB is essential to transform the data and ensure compatibility between the ONXX AI-Model and the IEC 61499 environment. This might involve converting data types or structures to match the requirements of the AI functions. For the prototype, the FLOAT data type is used.

- The communication between IEC 61499 Runtime and AI runtime takes place via OPC-UA or MQTT configuration, which is automatically added during the serialization of the model input and output data to the function block template, which considers the datatypes can be transferred with the right format.

5 Conclusions

Embedding of intelligence functionalities, such as machine learning models or any other type of AI, becomes the natural evolution empowering the distribution of control functions, also considering the bridge cloud-to-edge.

In the deliverable, we have seen how to embed AI functionalities and the best practice for developing FB - Function Block - in the IEC 61499.

- The document details the integration of an AI runtime environment within an IEC 61499 Edge Gateway, enhancing its capabilities for applications in industrial automation. The Key element is AI Runtime Integration: It provides the necessary infrastructure to execute AI models alongside control applications on the shop floor, with a focus on facilitating various communication protocols via the Embedded Edge Connectivity Layer (EECL).
- AI Components and Platforms: AI runtime environments encompass hardware acceleration, framework support, inference/training capabilities, model format compatibility, and specific optimizations for efficient AI model executions. AI runtimes can be installed on diverse platforms like PCs, local servers, public cloud, built-in devices, and containers.
- Model Conversion: The compatibility between different AI frameworks like TensorFlow and ONNX is addressed, allowing easier migration of deep learning models between platforms. The procedure for TensorFlow-to-ONNX conversion is provided.

- IEC 61499 Integration: Within the IEC 61499 platform, the goal revolves around capturing battery status from AMRs, allowing predictive monitoring for more informed control actions. The integration process encompasses importing AI functions/models, creating function blocks for control functions, utilizing communication protocols for interactions, and enabling data exchange between the control system and the AI runtime.
- IDE Plugin Implementation: The IDE plugin enhances the engineering tool, facilitating the integration of AI models into the IEC 61499 environment. It aids in creating XML representations of function blocks, handles data type mappings between ONNX and IEC 61499, and ensures proper communication configurations.

The primary use case involves Automated Mobile Robots (AMRs) in Reepack's food packaging production line. Predictive models are developed to improve battery management and ensure uninterrupted operation by analysing historical data on AMR battery usage.

In conclusion, integrating AI within the IEC 61499 environment represents a significant stride in industrial automation, fostering advanced capabilities in monitoring, decision-making, and optimizing operations, especially in applications like AMR battery management.

References

Aloqaily, M. J. (2019). *An IoT Cloud System for Traffic Monitoring and Vehicular Accidents Avoidance Based on Integration of Microservices and Kafka Streams.* Future Generation Computer Systems.

Cândido, G. C. (2013). *Service-Oriented Infrastructure to Support the Deployment of Evolvable Production Systems.* IEEE Transactions on Industrial Informatics, vol. 9, no. 3.

Cengic, G. &. (2018). *Embedding Python for Statistical Computing and Machine Learning in Open-Source IEC 61499 Automation and Control Tools.* IEEE 16th International Conference on Industrial Informatics (INDIN).

Khalgui, M. L.-A. (2018). *Reconfigurable Embedded Control Systems Applications for Industry 4.0.* Journal of Manufacturing Systems, vol. 46.

L. Wang, W. J. (2006). *Embedding machining features in function blocks for distributed process planning.* International Journal of Computer Integrated Manufacturing.

Lewis, R. M. (2016). *Developing Industrial Internet of Things applications with IEC 61499.* IEEE Industrial Electronics Magazine, vol. 10, no. 4.

Liu, X. L. (2019). *Edge Computing for Cyber Physical Systems: A Survey.* IEEE Access, vol. 7.

Mahmoud, Q. H. (2015). *A MQTT-based Mobile Cloud Middleware Framework for Automating Internet of Things Objects.* IEEE 2nd World Forum on Internet of Things (WF-IoT).

S. Campanelli, P. F. (n.d.). *Integration of existing IEC 61131-3 systems in an IEC 61499 distributed solution.* Proceedings of 2012 IEEE 17th International Conference on Emerging Technologies & Factory Automation (ETFA 2012), Krakow, Poland, 2012, pp. 1-8, doi: 10.1109/ETFA.2012.6489671.

Thramboulidis, K. &. (2015). *Towards an Integrated Reference Architecture for Cyber-Physical Systems of Systems.* IEEE 20th International Conference on Emerging Technologies and Factory Automation (ETFA).

Vyatkin, V. (2013). *Software Engineering in Industrial Automation: State-of-the-Art Review.* IEEE Transactions on Industrial Informatics.

Zoitl, A. &. (2010). *Applying Open Source Middleware Standards to the Integration of Industrial Automation and Manufacturing Execution Systems.* IEEE Conference on Emerging Technologies and Factory Automation.