Interoperability for Industrial Internet of Things Based on Service-oriented Architecture

An Ngoc Lam[1, 2], Øystein Haugen[1], and Jerker Delsing[2]

[1] *Faculty of Computer Sciences, Østfold University College,* Halden, Norway
[2] *Dept. of Computer Science, Electrical and Space Engineering, Luleå University of Technology,* Luleå, Sweden
Email: {an.n.lam,oystein.haugen}@hiof.no, jerker.delsing@ltu.se

Abstract—**The new Industry 4.0 envisions a future for agile and effective integration of the physical operational technologies (OT) and the cyber information technologies (IT) as well as autonomous cooperation among them. However, the wide variety and heterogeneity of industrial systems and field devices - especially on the factory floor - increase integration complexity. To address these challenges, new technologies and concepts such as the Industrial Internet of Things (IIoT), Service-oriented Architecture (SoA), Semantic Technologies, Machine Learning and Artificial Intelligence are being introduced to the industrial environment. In this paper, we focus on how industrial automation systems and field devices can be integrated into the IIoT framework and coordinated to adapt to dynamic operating environment. Specifically, this paper proposed an interoperability solution that makes use of SoA and Semantic Technologies to achieve supervised coordination of IIoT application systems. To illustrate the potential of this approach, the Service-oriented Architecture-based Arrowhead Framework is used as the fundamental framework for the implementation of the approach.**

I. Introduction

The development of new technologies such as Industrial IoT (IIoT), Industrial Cyber-Physical Systems (ICPS), and Service-oriented Architecture (SoA) has opened a promising future where the physical and the digital worlds work together, leading to a new wave of industrial revolution, namely Industry 4.0 [1]. Industry 4.0 envisions an industrial environment where field devices that employ different physical operation technologies (OT) are represented digitally in the cyber information technology (IT) for autonomously monitoring, performing control, and making management decisions [2]. The two primary goals to enable this vision are cross-layer integration, and semantic interoperability among the components and systems with open vendor-independent architecture [3].

Consequently, there is a high demand for interoperability among the processes, devices from the factory floor, and enterprise and business systems. The main goal is to enable the high-level application to exploit process data produced in the field. For example, these data can be provided to the Product Lifecyle Management (PLM) system for real-time coordination of the production or the Enterprise Resource Planning (ERP) system to update the production plans and schedules. Furthermore, the field data is also critical for Manufacturing Execution Systems (MES) and Supervisory Control and Data Acquisition (SCADA) systems. However, due to the high complexity and variety of heterogeneous devices and technologies used in the field, the integration between these systems and the factory floor is often not completely achieved [2]. Furthermore, the communication protocols used by the field devices are not designed to fully support exchanging data with the standard IT systems. Instead, many current machines and devices in legacy systems are using vendor-dependent protocols with their own payload and syntaxes, leading to considerable integration efforts for manual configuration, parsing, and annotating the data [2].

SoA stands out as a promising candidate to support IT-OT integration in order to make the systems from these two layers more interoperable. There are several European projects such as SOCRADES [4], [5], SIRENA [6], SODA [7] that introduce a SoA middleware to exposes shop floor devices as services to the application at higher layer such as MES and ERP systems [8]. However, these frameworks are designed to support devices that can host web services, which is not the case for most legacy industrial systems.

To address these emerging integration challenges discussed above, this paper proposes an approach that facilitates the collaboration between physical devices and higher-level application systems by introducing an intermediate layer that transforms the data and operations at the OT layer into the services supported by the IT systems. To implement the proposed approach, the Arrowhead Framework [9] - a SoA cloud-based framework for IIoT - is applied as the fundamental technology. Furthermore, to coordinate the collaboration of the application systems, a supervision approach is developed by using semantic technologies to enable self-adaptation. The ultimate goal is to reduce the engineering cost related to the migration of legacy industrial systems while not introducing too much complexity to the design of those systems. We make the following contributions:

1) A service-oriented solution to integrate heterogeneous automation systems into an IIoT framework.
2) An edge-based supervision approach to make industrial IoT systems interoperable.
3) A prototype based on Arrowhead Framework is implemented to illustrate the great potential of the approach.

The remainder of this paper is organized as follows: Section II gives a brief overview of the Arrowhead Framework and the core systems that are used for the implementation of this approach. The proposed architecture for implementing interoperability between cyber (IT) and physical (OT) systems is discussed in Section III. In Section IV, the detail of the implementation is presented. Section V introduces a small use case of the proposed approach. In Section VI, we provide an overview of the related works and discuss the improvement of this approach toward those works. Finally, the paper is rounded up with conclusions and future works in Section VII.

II. Arrowhead Framework

Arrowhead Framework is a cloud-based framework that aims to facilitate the creation of *Local Automation Clouds* and support real-time, security, interoperability, and scalability requirements of automation and industrial IoT systems [9]. Arrowhead Framework approaches the transformation from large monolithic organizations to multi-stakeholder cooperations via the interactions of multiple Local Automation Clouds, thus satisfying the high-level requirements of today's society such as sustainability, flexibility, efficiency, and competitiveness [10]. Within Arrowhead Framework,

- A *service* contains the information about the interface (e.g., communication protocol, data format, connection endpoint) that is used for information exchanging among the application systems.
- A *system* is a software artifact that either produces or consumes services. There are three different types of systems: *the mandatory core systems* that empower the fundamental properties of a local cloud, *the automation support core systems* that facilitate the automation application design, engineering, and operation, and *the application systems* that implement the application functionalities.
- A *device* is a piece of equipment, machine, or hardware with computational and memory capabilities which can deploy one or several systems.

By employing the SoA, within one Arrowhead Local Automation Cloud, the communication between the systems becomes producing and consuming services, thus constituting a System of Systems Architecture. This architecture introduces the following properties: (i) loose coupling, which enables the producer to decide whom to consume its services, (ii) late binding, which makes the consumer connect to the services at any time, and (iii) lookup for run-time service discovery [11].

To construct a minimal Arrowhead Local Automation Cloud, the following three mandatory core systems are needed:

- The *Service Registry* stores the information of the services published within the local cloud. The Service Registry produces services for the provider systems to register their services as well as for the consumer systems to look for published services in the local cloud.
- The *Authorization System* provides services for authenticating and authorizing application service usage.
- The *Orchestration System* provides mechanism for distributing instruction about the communication between the consumer and provider systems.

Besides those mandatory core systems, there are also various automation support core systems that facilitate the design and implementation of automation systems such as the *System Registry* and *Device Registry* that support the on-boarding procedure of new systems and devices [10], the *Configuration System* that provides services for the distribution of configurations to the application systems [12], the *Data Manager* that provides services for short-term and long-term data storage, data retrieval and filtering, the *Autonomic Orchestration System* that enables self-adaptation at service level by using semantic rules and ontological reasoning [13].

III. The Proposed Architecture

Traditional industrial architectures are developed based on a pyramid model with various layers organized in a hierarchical structure, which have very limited support for vertical integration between the layers [2]. As a result, several alternative architectures which have less levels, stricter boundary and more standard inter-connected components have been proposed. Typically, these modern architectures are organized based on two layers: OT and IT layers, where the communication protocols employed by the OT systems are usually different from the information models used in the IT domain. As a result, there is a need of an intermediate layer that facilitates the transformation of data of the physical devices and components into the information models applied by the systems of the cyber layer.

This paper proposes a solution for (i) integrating the legacy industrial devices into application systems at cyber layer and for (ii) the interoperability among those systems. First of all, for the integration support, we employ the SoA architecture in which the data of physical devices will be abstracted into services that are made available for the application systems to consume. Figure 1 illustrates the components necessary for the implementation of the approach. Accordingly, as most legacy industrial devices do not support modern communication protocols (e.g., OPC-UA, RESTful, MQTT) which are usually employed by the application systems at the high-level layer, we propose an intermediate layer between the physical devices and application systems, namely *Adapter Systems*, to transform the data and functionalities of the physical devices into services. As a result, this integration step is done separately from the physical layer, thus there is no need for modifications or upgrades of physical devices. Furthermore, once the physical services are published, the application systems can decide which physical services to consume at any time in a flexible way. However, this approach requires the implementation of an adapter for each legacy communication protocol. Nevertheless, the resulting cost related to this migration step may not be high as there may be a limited number of legacy protocols used in a particular manufacturing system and the adapter can be reused and duplicated for the same protocol.

Secondly, in order to make the application systems interoperable, we apply a supervision approach enabled by the *Interoperability Management Systems*. Particularly, those Interoperability Management Systems are deployed at the edge of the network to enable real-time monitoring and coordinating the collaborations among the application systems. For example, these Interoperability Management Systems can provide suggestions for an application system to connect to appropriate service or to react to an anomaly that is not handled in their designs. The Interoperability Management Systems do not handle any application logic; instead, they monitor the application systems and provide them necessary instructions. These instructions are made based on the high-level policies which could be added or updated at run-time. Therefore, this approach increases the flexibility and possibilities for the collaboration among the application systems. Furthermore, as the supervision logic is implemented separately in the Interoperability Management layer, this edge-based supervision approach helps to reduce the complexity of the application systems as well as the effort related to maintenance and diagnosis of the unexpected situations.

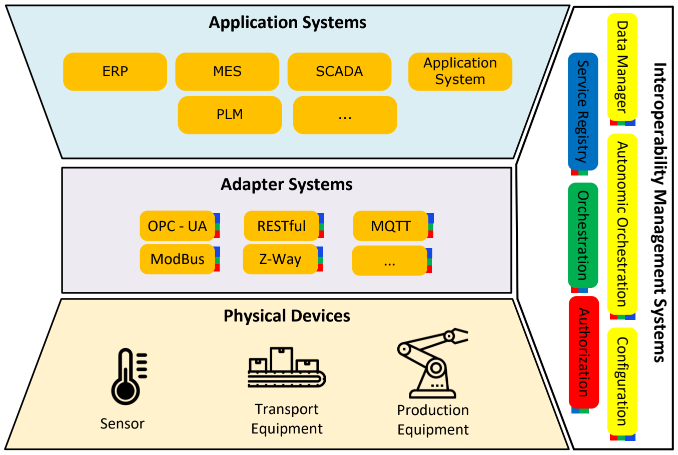

Fig. 1. Implemented components of a cross-layer architectures for enabling interoperability among IIoT systems.

IV. Implementation Approach

This section discusses the implementation of our proposed approach in detail. Specifically, we employ the Arrowhead Framework as the fundamental technology for the implementation of our service-oriented approach. As can be seen from Figure 1, we exploit existing core systems and application systems for the implementation of the Adapter and the Interoperability Management Systems. Figure 2 illustrates the communication flow of those systems within an Arrowhead Local Cloud.

For the adapters, Arrowhead Framework provides the client library that facilitates the development of Arrowhead application systems that publish RESTFul services in a flexible and efficient way [14]. This library is proven to reduce code duplication and enhance the readability of the implementation. This library can also help to simplify the implementation of the adapters for other protocols as well. As an example, the work in [15] presented an engineering procedure to wrap OPC-UA functions and publish them as a service to the Arrowhead

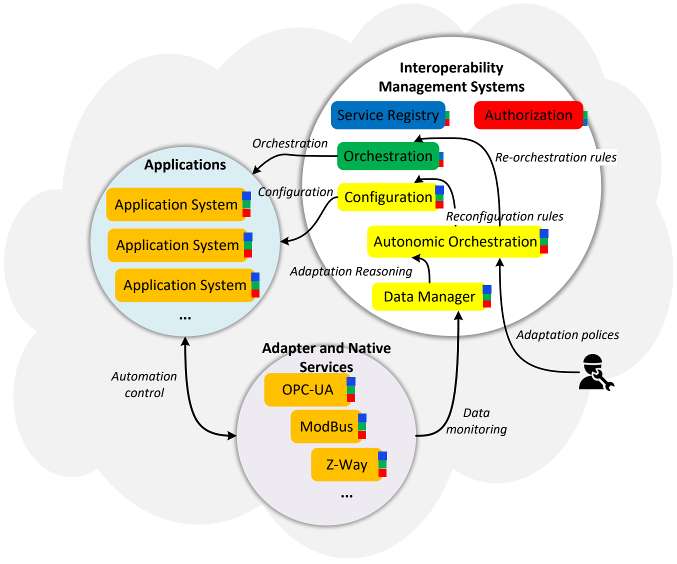

Fig. 2. Communication flow of the systems within an Arrowhead Local Automation Cloud.

local cloud. For other industrial protocols, the adapters can be manually implemented with the client library based on the same mechanism proposed in [15].

Furthermore, within Arrowhead Framework, there are also available native solutions to integrate industrial standards (e.g., Modbus TCP, MQTT, Z-Way, FIWARE and BaSys) and make them available under the form of services. For example, the *Translation System*, an automation support core system, provides translation as a service that can map OPC-UA to other protocols such as CoAP, RESTful, MQTT [9], [16]. Arrowhead Framework also offers another automation support core system, namely *Modbus TCP System*, that provides the mechanism to wrap the Modbus TCP functions to RESTful service. As discussed earlier, the adapters and these native support systems can be reused for other physical devices employing the same protocol, thus saving effort for the implementation in this step.

Regarding the Interoperability Management Systems, the following systems are leveraged to enable interoperability among the application systems:

- *Service Registry* and *Authorization System*: these two systems provide services for the other systems to discover, authenticate and authorize the service usages. Typically, the interaction with these two core systems will be handled automatically by the *Orchestration System* whenever an application system makes an orchestration query. Therefore, application systems do not need to consume the services provided by these two core systems. Detail of the interaction is discussed in [9] and is not presented in figure 2 for simplicity of the illustration.
- *Data Manager System*: acts as a central storage of the data from the services for monitoring purposes. Particularly, all of the application systems that want to collaborate with each other have to record their data to the *Data Manager*.

```
(?c rdf:type auto:Consumer)
(?c auto:consumesService ?s1)

(?s1 auto:hasState auto:OfflineState)
(?p1 auto:producesService ?s1)
(?d1 auto:hasService ?s1)
(?o1 sosa:madeBySensor ?d1)
(?o1 auto:hasUnit ?u1)

(?s2 auto:hasState auto:OnlineState)
(?p2 auto:producesService ?s2)
(?d2 auto:hasService ?s2)
(?o2 sosa:madeBySensor ?d2)
(?o2 auto:hasUnit ?u2)

notEqual(?u1 ?u2)
->
substitute(?c ?s1 ?p1 ?s2 ?p2)
configure(?c 'unit' ?u2)
```

Fig. 3. An adaptation policy to substitute one temperature service that is disconnected and reconfigure the unit.

To do so, these application systems need to consume the *Proxy Service* provided by the *Data Manager*. This service provides a mechanism to cache one message and is suitable for low-power, resource-constrained devices. By employing the *Data Manager* as the central data storage, we avoid creating many connections to the application services as well as reduce the effort related to the discovery of necessary services for monitoring and the transformation of the data.

- *Autonomic Orchestration System*: this is the main system that implements the centralized supervision approach for interoperability. Specifically, this system is developed based on our previous work that applied semantic technologies to standardize the data and further reason for smart decision on the structured data [17]. The final goal is to address semantic interoperability issues and enable self-management for the application systems. As can be seen from figure 2, the *Autonomic Orchestration System* also consumes the *Proxy Service* to monitor relevant physical devices and reason for any adaptations specified by the *Adaptation policies*. These adaptations will then be transform into *reconfiguration rules* and *re-orchestration rules* to supervise the interaction of the corresponding application systems. Specifically, the *Orchestration Updater Service* provided by the *Autonomic Orchestration System* is used to handle this transformation and push the new orchestration/configuration rules to the *Orchestration System* and *Configuration System* respectively. *Adaptation policies* specify the high-level interaction strategies that show the application systems which and how to interact with each other. In this approach, we use semantic rule as the syntax to specify these policies. Figure 3 presents an example of such policies. This polices guides the consumer system to substitute its currently-consuming service (i.e., the *substitute* predicate) that is disconnected from the cloud (i.e., has state *OfflineState*) by another relevant service (i.e., has state *OnlineState*) and reconfigure the unit (i.e., the *configure* predicate) if the units are not the same (i.e., the *notEqual* predicate). The *Adaptation policies* can be registered to the *Autonomic Orchestration System* manually by the administrator of the local cloud or programmatically by the application systems via the *Orchestration Register Service* provided by this core system. More information about the implementation of the *Autonomic Orchestration System* can be found in [13].
- *Configuration System*: this system stores the configuration and settings of the application systems. Whenever there is newly-generated *reconfiguration rules*, the *Autonomic Orchestration System* will push these rules to the *Configuration System* via *Config Management Service* provided by this *Configuration System*. Subsequently, the new *Configuration* can be retrieved by the application systems via the *Config Service* also provided by this core system.
- *Orchestration System*: this system stores the orchestration rules of the application systems and provide them to the application system on request. Similar with the *Configuration System*, the *Orchestration System* will receive new *re-orchestration rules* from the *Autonomic Orchestration System* via *Orchestration Store Management Service* and push them to the corresponding application systems via *Orchestration Service*.

V. Application Example

In the previous section, we have presented our approach to enabling interoperability among the IIoT systems by exploiting different core systems of the Arrowhead Framework. In this section, in order to illustrate the detailed interaction among those systems, we present a step-by-step communication procedure within an example application that displays the temperature value and automatically reconfigures itself whenever errors occur. In this example, there are three application systems involved: *Temperature Display System*, *Temperature1* and *Temperature2*. The two temperature systems capture the measurements from two temperature sensors and provide services for other systems to retrieve those values. The two sensors are located at the same location but configured to provide values in two different units: Fahrenheit and Celsius. Figure 4 shows the output of the service provided by *Temperature1*. The services employ SenML[1] format to display the information such as name of the provider system, timestamp, unit, and value of the measurement. The *Temperature Display System* is a web application that consumes either one of those two temperature services and displays the temperature in Kelvin degree to a web page. Internally, this system is implemented with two functions to convert from Fahrenheit or Celsius to Kelvin. In this example, by employing our approach, the *Temperature Display System* is enabled to re-orchestrate to use a new temperature service and reconfigure the conversion function accordingly based on the collaboration with the Interoperability Management Systems.

[1]https://datatracker.ietf.org/doc/html/rfc8428

```
{
  "bn" : "Temperature1",
  "bt" : 1.627410726679E12,
  "bu" : "Fahrenheit",
  "ver" : 1,
  "e" : [ {
    "n" : "Indoor Temperature",
    "v" : 107.08383965661564,
    "t" : 1.627410726679E12
  } ]
}
```

Fig. 4. Payload of the Temperature1 service.

illlFigure 5 illustrates the interaction of the systems in this example. Initially, the *Temperature Display System* is configured to consume the service of *Temperature1*. Message #1 indicates the main operation of the *Temperature Display System*, which is periodically retrieving the temperature value via the service, convert it to Kelvin degree, and display it accordingly. Furthermore, this system may register its orchestration and configuration policies with the Autonomic Orchestration System via the *Orchestration Register Service* (message #2). In this example, we use only one policy which is shown in figure 3. Specifically, this policy guides the *Temperature Display System* to re-orchestrate to the other temperature service and re-configure the unit conversion function respectively whenever the currently-consuming temperature service disconnected. Moreover, the two temperature services need also to update their data to the Data Manager System periodically via *Proxy* service in order for the Autonomic Orchestration System to monitor their status (message #3-6). In this example, the Autonomic Orchestration System is configured to change the status of the services to *OfflineState* if their measurement values are not updated within 5 seconds. Once this situation happens, new orchestration and configuration will be generated based on the registered policy in figure 3 and be stored in the Orchestration System (via *Orchestration Store Management Service*) and the Configuration System (via *Config Management Service*) respectively (message #7, 8). Then, this new adaptation will be updated to the *Temperature Display System* for re-orchestration and re-configuration (message #9, 10). Finally, the *Temperature Display System* continues its operation with the newly-configured temperature service (message #11).

VI. Related Work

Due to the growing demand for interoperability among the processes, devices, and systems - down from the factory floor up to the enterprise systems - the integration of IT and OT layers is of utmost importance [18]. However, due to the wide variety and heterogeneity of the technologies and standards used on the factory floor, this integration becomes a significant challenge. The most common solution is to develop an intermediate layer to transform the heterogeneous protocols at the OT layer into a unified information model supported by the IT. For example, Schmidtmann et al. proposed a hardware abstraction layer for uniformly mapping the physical devices of the manufacturing system into symbols in XML configuration files [19]. These XML files will then be processed by the layer whenever there are actions performed by these devices. Givehchi et al. proposed a similar approach with an interoperability layer that maps the data of the physical layer into OPC-UA information model [2]. These related works require the business systems to adapt to the designated protocols (e.g., OPC-UA) in order to interact with the physical layer, which may require substantial modification of the application systems. Also, bottlenecks issues may be experienced at the intermediate layer whenever the number of physical devices increases. By introducing an adapter for each communication protocol, our approach can avoid too much modification of the application system while also addressing the scalability issue as those adapters can be deployed on distributed hardware.

SoA is also a potential architecture for the heterogeneity challenge of the physical layer. The use of SoA in automated manufacturing is being investigated in several European projects such as SOCRADES [4], [5], SIRENA [6], SODA [7]. The primary objective of the SOCRADES project is to develop an architecture for the future factory including middleware that supports the devices from the shop floors. SIRENA and SODA project proceeds along similar lines that introducing SoA frameworks for multiple domains apart from industrial automation. These projects are developed based on the smart devices that can host web services (e.g., DPWS enabled devices (Devices Profile for Web Services)), which demands device modifications as most legacy industrial devices support no protocol stack for modern protocols. In this work, we address this problem by introducing a separate layer handling the mapping of those devices to services.

Furthermore, in this work, we move one step further by monitoring the services and providing instructions for re-orchestration and reconfiguration with the support of semantic technologies. Similar support can be found in [5], [20], [21]. However, those related works are either at the conceptual stage or addressed only specific application domains.

VII. Conclusion

Service-oriented concepts enable the collaboration of heterogeneous hardware and software platforms, which is crucial to the future industrial revolution. In this paper, by applying the SoA, we proposed an approach to enable the integration of the physical devices of the OT associated with the technical process into the application systems from the IT associated to the business process. Furthermore, we also present our supervision solution to enable the interoperability of those application systems. The approach is implemented by leveraging different core systems provided by the service-oriented Arrowhead Framework. This proposed implementation can help to increase the efficiency for the enterprise as it improves the interoperability without any modification to the physical devices while not introducing too much complexity to the high-level enterprise systems.

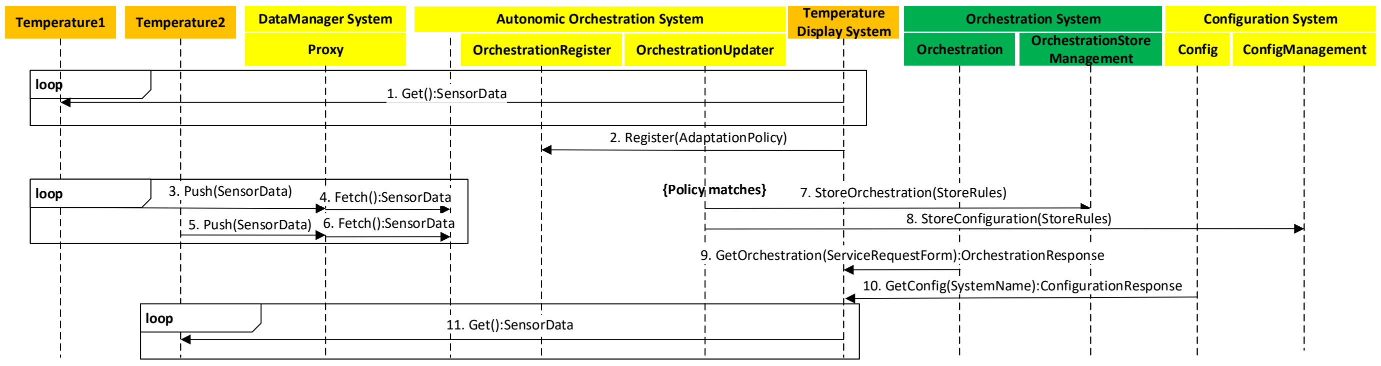

Fig. 5. Interaction among the systems of the example application.

As future work, we will evaluate additional requirements for the security and usability of the proposed implementation. Specifically, other automation support core systems such as System Registry and Certificate Authority will be investigated to enable seamless integration and monitoring of new application systems and address advanced security requirements related to the usage of services. Furthermore, an approach to assist the user in creating, editing, and validating adaptation policies will be investigated as the current syntax of the semantic rule is very error-prone.

ACKNOWLEDGMENT

Research leading to these results has received funding from the EU ECSEL Joint Undertaking under grant agreement n° 737459 (Productive4.0 project) and n° 826452 (Arrowhead Tools project), and from the Research Council of Norway (project n° 282904).

REFERENCES

[1] H. Lasi, P. Fettke, H.-G. Kemper, T. Feld, and M. Hoffmann, "Industry 4.0," *Business & information systems engineering*, vol. 6, no. 4, pp. 239–242, 2014.
[2] O. Givehchi, K. Landsdorf, P. Simoens, and A. W. Colombo, "Interoperability for industrial cyber-physical systems: An approach for legacy systems," *IEEE Transactions on Industrial Informatics*, vol. 13, no. 6, pp. 3370–3378, 2017.
[3] K. Nagorny, A. W. Colombo, and U. Schmidtmann, "A service- and multi-agent-oriented manufacturing automation architecture: An iec 62264 level 2 compliant implementation," *Computers in Industry*, vol. 63, no. 8, pp. 813–823, 2012.
[4] L. M. S. De Souza, P. Spiess, D. Guinard, M. Köhler, S. Karnouskos, and D. Savio, "Socrades: A web service based shop floor integration infrastructure," in *The internet of things*. Springer, 2008, pp. 50–67.
[5] A.-W. Colombo, S. Karnouskos, and J.-M. Mendes, "Factory of the future: A service-oriented system of modular, dynamic reconfigurable and collaborative systems," in *Artificial intelligence techniques for networked manufacturing enterprises management*. Springer, 2010, pp. 459–481.
[6] H. Bohn, A. Bobek, and F. Golatowski, "Sirena-service infrastructure for real-time embedded networked devices: A service oriented framework for different domains," in *International Conference on Networking, International Conference on Systems and International Conference on Mobile Communications and Learning Technologies (ICNICONSMCL'06)*. IEEE, 2006, pp. 43–43.
[7] S. De Deugd, R. Carroll, K. Kelly, B. Millett, and J. Ricker, "Soda: Service oriented device architecture," *IEEE Pervasive Computing*, vol. 5, no. 3, pp. 94–96, 2006.
[8] J. Delsing, J. Eliasson, R. Kyusakov, A. W. Colombo, F. Jammes, J. Nessaether, S. Karnouskos, and C. Diedrich, "A migration approach towards a soa-based next generation process control and monitoring," in *IECON 2011-37th Annual Conference of the IEEE Industrial Electronics Society*. IEEE, 2011, pp. 4472–4477.
[9] P. Varga, F. Blomstedt, L. L. Ferreira, J. Eliasson, M. Johansson, J. Delsing, and I. M. de Soria, "Making system of systems interoperable–the core components of the arrowhead framework," *Journal of Network and Computer Applications*, vol. 81, pp. 85–95, 2017.
[10] A. Bicaku, S. Maksuti, C. Hegedűs, M. Tauber, J. Delsing, and J. Eliasson, "Interacting with the arrowhead local cloud: On-boarding procedure," in *2018 IEEE industrial cyber-physical systems (ICPS)*. IEEE, 2018, pp. 743–748.
[11] J. Delsing, J. Eliasson, J. van Deventer, H. Derhamy, and P. Varga, "Enabling iot automation using local clouds," in *2016 IEEE 3rd World Forum on Internet of Things (WF-IoT)*. IEEE, 2016, pp. 502–507.
[12] O. Carlsson, P. P. Pereira, J. Eliasson, J. Delsing, B. Ahmad, R. Harrison, and O. Jansson, "Configuration service in cloud based automation systems," in *IECON 2016-42nd Annual Conference of the IEEE Industrial Electronics Society*. IEEE, 2016, pp. 5238–5245.
[13] A. N. Lam and Ø. Haugen, "Applying semantics into service-oriented iot framework," in *2019 IEEE 17th International Conference on Industrial Informatics (INDIN)*, vol. 1. IEEE, 2019, pp. 206–213.
[14] T. Pedersen, M. Albano, and B. Nielsen, "Reengineering the lifecycle of arrowhead applications: From skeletons to the client library," in *IECON 2019-45th Annual Conference of the IEEE Industrial Electronics Society*, vol. 1. IEEE, 2019, pp. 5519–5524.
[15] A. N. Lam and Ø. Haugen, "Implementing opc-ua services for industrial cyber-physical systems in service-oriented architecture," in *IECON 2019-45th Annual Conference of the IEEE Industrial Electronics Society*, vol. 1. IEEE, 2019, pp. 5486–5492.
[16] H. Derhamy, J. Rönnholm, J. Delsing, J. Eliasson, and J. van Deventer, "Protocol interoperability of opc ua in service oriented architectures," in *2017 IEEE 15th International Conference on Industrial Informatics (INDIN)*. IEEE, 2017, pp. 44–50.
[17] A. N. Lam and Ø. Haugen, "Supporting iot semantic interoperability with autonomic computing," in *2018 IEEE Industrial Cyber-Physical Systems (ICPS)*. IEEE, 2018, pp. 761–767.
[18] D. R. Harp and B. Gregory-Brown, "It/ot convergence bridging the divide," *NexDefense*, p. 23, 2015.
[19] U. Schmidtmann, G. Kreutz, M. Barkhoff, K. Virkus, T. Stöckmann, and M. Jovic, "Microprogrammable hardware abstraction layer for flexible automation," in *2009 IEEE Conference on Emerging Technologies & Factory Automation*. IEEE, 2009, pp. 1–7.
[20] J. Kiljander, A. D'elia, F. Morandi, P. Hyttinen, J. Takalo-Mattila, A. Ylisaukko-Oja, J.-P. Soininen, and T. S. Cinotti, "Semantic interoperability architecture for pervasive computing and internet of things," *IEEE access*, vol. 2, pp. 856–873, 2014.
[21] S. Poslad, S. E. Middleton, F. Chaves, R. Tao, O. Necmioglu, and U. Bügel, "A semantic iot early warning system for natural environment crisis management," *IEEE Transactions on Emerging Topics in Computing*, vol. 3, no. 2, pp. 246–257, 2015.