Software Toolkit for Development of Interoperable Communications in Data-driven Systems

Petri Kannisto [*] **Antti Kätkytniemi** [*] **Matti Vilkko** [**] **David Hästbacka** [*]

[*] *Faculty of Information Technology and Communication Sciences, P.O. box 553, 33014 Tampere University, Finland (e-mail: petri.kannisto/antti.katkytniemi/david.hastbacka@tuni.fi)*
[**] *Faculty of Engineering and Natural Sciences, P.O. box 589, 33014 Tampere University, Finland (e-mail: matti.vilkko@tuni.fi)*

Abstract: The ease of software development, maintenance, commissioning and reconfiguration is paramount in industrial systems, because these must be adaptable to changes in customer demand and the evolution of production assets. Such adaptations are a core motivator in the fourth industrial revolution. Additionally, software applications should be able to communicate over the logical and geographical borders of production plants, regardless of software platforms and vendors. However, these issues remains unaddressed in current software tools, although a solution is reachable with a proper software toolkit. As a solution, this article introduces such a toolkit to support the development of industrial software systems. The toolkit implements a communication stack with both a communication protocol and an information model. The long-term goal is to enable all communication in industrial plants with such a toolkit regardless of software products, vendors and platforms. Additionally, the toolkit provides auxiliary tools to help in the development of industrial software, such as services to provide historical data for simulation purposes. As an outcome, the toolkit provides insight about the development needs of industrial communications.

Keywords: Message-oriented Middleware, Systems Integration, Service-oriented Architecture, Loose Coupling, Scalability, Industrial Cyber-physical Systems, Copper Smelting, Steel Production, Online Life Cycle Assessment

1. INTRODUCTION

The industrial community lacks software tools for the development of communications in Industry 4.0 (I4.0). Among other goals, I4.0 aims at interoperability in a multi-vendor environment as well as reconfigurability and scalability to growing data volumes in communication (Panetto et al., 2019). To reach these goals, there are fundamental technologies, but these are merely basic components that lack supportive tools for development. For instance, OPC UA PubSub (2017) aims at combining both scalability and interoperability but lacks an architecture for a plant-wide communication system. Therefore, unaddressed issues remain, as studied in the project *Coordinating Optimisation of Complex Industrial Processes* (COCOP, 2020).

To solve the challenges of I4.0 in communications, this paper presents the *COCOP Toolkit*. The first part of the toolkit implements a communication stack as reusable Application Programming Interfaces (API). The stack is based on a common information model and a message-oriented middleware as the communication protocol (AMQP 0-9-1; Advanced Message Queueing Protocol). With the APIs, the software developers can concentrate on application logic rather than communication-related details while still maintaining a scalable, interoperable system architecture. AMQP has been designed for growing message volumes and even supports load balancing. Therefore, the toolkit contributes to data-driven systems, which are in the core of *smart production* (Cheng et al., 2018). The second part of the toolkit consists of auxiliary tools that help in application development in industrial context. These include, for instance, a data source simulator and a message logger.

This paper has the following research objective:

Present a software toolkit for the development of loosely coupled, scalable, interoperable industrial systems, backed with proofs of concept

[⋆] This project has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement No 723661. This study reflects only the authors' views, and the Commission is not responsible for any use that may be made of the information contained therein. The authors want to express their sincere gratitude to the funder and the project partners in COCOP project (Coordinating Optimisation of Complex Industrial Processes, `https://www.cocop-spire.eu/`). Additionally, this work was supported by the Academy of Finland (grant 310098).

Next, Section 2 overviews the background of the research. Section 3 introduces the software toolkit followed by proofs of concept in Section 4. Finally, Section 5 discusses and concludes the paper.

2. BACKGROUND

Earlier projects have suggested architectures for either interoperability or data-driven scenarios but not for both in the scope of plant-wide systems. de Souza et al. (2008) propose a Web-Service-based middleware called *Socrades*, but this is restricted to devices and the applied technologies are less concerned with scalability. Karnouskos et al. (2014) introduce *IMC-AESOP*, a framework to realise a 'service cloud' for the needs of production systems, but this lacks a communication medium. Theorin et al. (2017) propose a message-bus-based architecture *LISA* that focuses on equipment rather than the plant-wide scope, and there is no focus on applying existing standards in communications. *MONSOON* by Sarnovsky et al. (2018) aims at facilitating the processing of industrial big data but omits the goal to reduce heterogeneity in communications. Respectively in *DISIRE*, Goldin et al. (2017) focus on data rather than interoperability. Finally, Gosewehr et al. (2017) propose the *PERFORM* middleware with a common data model but no attempt to unify the communication protocol.

First released in 2006, Open Platform Communications Unified Architecture (OPC UA, 2017) aims at enabling interoperability within industrial plants with a series of standards for multiple aspects in communication. OPC UA is supported in multiple software environments, and there is a selection of tools from a number of vendors (OPC Products, 2020). As the conventional client-server approach lacks loose coupling and scalability to data-intensive applications, OPC UA PubSub (2017) has been released. However, from the viewpoint of plant-wide communications, an essential shortcoming is the lack of message routing. Therefore, the current PubSub mainly provides a data link rather than an all-round communication solution, which is a goal in COCOP project.

Fig. 1 illustrates the system architecture specified in COCOP project. The architecture aims at connecting all systems in the plant-wide scope with a single message bus. The message bus enables interoperability with two aspects: a single communication protocol and commonly agreed information model. First, as the concrete protocol, AMQP 0-9-1 (2008) has been chosen, as the newer AMQP 1.0 lacks a support for message routing. From the beginning, AMQP was designed for a data-intensive environment (O'Hara, 2007). In COCOP, the AMQP implementation is *RabbitMQ*, which is an open-source product. The second aspect, information model, is based on the work of two standardisation bodies that provide data structures in Extensible Markup Language (XML). Generic data is serialised using standards from the Open Geospatial Consortium (OGC, 2020). *Observations and Measurements*, *TimeseriesML* and *Sensor Web Enablement Common Data Model*, among others, enable the delivery of industrial data. Respectively, production schedules are serialised in the ANSI/ISA-95-based *Business to Manufacturing Markup Language* (B2MML, 2013).

Earlier COCOP publications have elaborated the COCOP concept (Kannisto and Hästbacka, 2018) and explained work-in-progress about the information model (Hästbacka et al., 2018b).

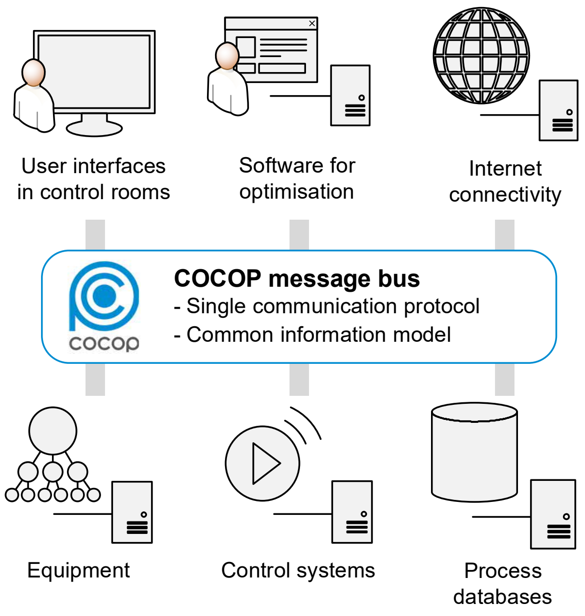
User interfaces in control rooms
Software for optimisation
Internet connectivity
COCOP message bus
- Single communication protocol
- Common information model
cocop
Equipment
Control systems
Process databases

Fig. 1: COCOP Architecture enables plant-wide interoperability with a message bus.

COCOP Toolkit is different from Enterprise Service Bus (ESB) because ESBs implement not only communication but also the adapters required for various communication protocols and data models that the connected software systems support (Boyd et al., 2008). Consequently, ESBs tend to contain business logic, which adds difficulty to maintenance by mixing the role of software solutions and the channel of communication.

3. SOFTWARE TOOLKIT

The COCOP message bus only requires a support for one communication protocol and certain information models to enable communications. However, the tasks of software development introduces issues that can be relieved with appropriate tools, as explained in this section.

3.1 Message Serialisation

A common information model is essential for interoperability, but its serialisation requires effort that could be re-used between applications if these run on the same platform. Thus, libraries were developed.

Cocop.MessageSerialiser.Meas API implements a data model based on selected standards from the OGC (2020). These include *Observations and Measurements* for metadata, *Sensor Web Enablement Common Data Model* and *Geography Markup Language* for measurements, data records and other data types as well as *TimeseriesML* for timeseries. Additionally, *Sensor Observation Service* enables the request-response communication of data, whereas *Sensor Planning Service* provides structures for the remote control of long-running tasks. The API has been implemented for .NET and Java.

Respectively, the API *Cocop.MessageSerialiser.Biz* implements structures for production schedules. There structures come from B2MML (2013) based on the standard *ANSI/ISA-95*. The API exists for .NET and Java.

3.2 Request-Response and Publish-Subscribe

The communication scenarios sometimes need a *request-response* approach and sometimes *publish-subscribe*. Request-response means that a piece of data is explicitly requested for as needed ('pull'), whereas publish-subscribe means that the data source informs any interested parties about changes in data ('push'). Of these, publish-subscribe is native in AMQP. Request-response can be built on top of it, but this requires additional effort that should be reusable between applications.

To re-use the implementation of request-response over AMQP, *Cocop.AmqpRequestResponseHelper* API was implemented. The API exists in C#/.NET and Java. To implement request-response, the developers could instead use Hypertext-Transfer-Protocol-based (HTTP) web services, but this would require a parallel communication channel. Furthermore, HTTP would require a mechanism to manage host addresses in the network, whereas the AMQP-based request-response implementation only needs topic names in the RabbitMQ server.

3.3 Connectivity of Optimisation Tools

In a plant-wide systems, optimisation tools must be connected to the network. AMQP 0-9-1 has client APIs for multiple platforms, but this excludes the platforms made for mathematics. In COCOP, there is a need to connect Matlab® with the message bus.

AMQP Math Tool Connector is an API for AMQP connectivity from Matlab. The API was developed in Java and can execute in Matlab as such. Therefore, the API connects Matlab algorithms and models with any software via AMQP. Furthermore, the Java implementation of *Cocop.MessageSerialiser* is functional in Matlab, which enables a full compliance with COCOP message bus.

3.4 Testing and Debugging

To enable safe, secure and efficient operation, it is essential to discover errors in software as early as possible. In software development, 'testing' refers to a systematic process of error detection. Testing should be applied even to data-intensive tools that optimise plant operation. However, such testing can be difficult especially if the tools are stateful and must observe long time periods of production to be operable. Testing can occur even online in the real environment, but this hampers observations from untypical conditions. Besides, this is slow if the production dynamics is slow, and the repeatability of tests is restricted. Therefore, there should be tools for early testing with offline data.

To enable offline testing, COCOP delivered the web services *Process Data Simulator* and *Requested History Data Publisher*. Process Data Simulator reads earlier recorded data from a spreadsheet file (Comma Separated Values) and saves the values to a local database. Then, it can publish these values to the message bus as if they came from an online production system, which enables the testing of simulation tools. The same tests can run as many times as needed, so the tools under test can be fine tuned. Process Data Simulator can even accelerate clock time, which makes testing faster if the related production processes have slow dynamics. The application has a Rest interface for remote operation. Respectively, Requested History Data Publisher serves past data in the request-response pattern. This enables the population of values to applications that otherwise receive data with publish-subscribe. With publish-subscribe, such population is slow if the frequency of data publishing is low. For instance, if the state of a process changes only a few times per hour, it is better to retrieve the current state explicitly as applications start up.

Furthermore, in a system of systems, there should be tools to monitor how the systems interact, as this helps in resolving errors not only during development but also in commissioning. Therefore, a message-oriented environment should have a message log. For this need, the desktop application *Message Logger* was developed. Message Logger can monitor all traffic in the message bus, store these messages in the hard disk and display these.

3.5 Relationship of Items

Table 1 summarises the items of the toolkit. The source code of each piece of software has been opened in Github [1]. Therefore, these are available for anyone to exploit or even develop further.

Fig. 2 shows how the toolkit items relate to COCOP message bus. The *MessageSerialiser* APIs implement the information model for more convenient access in software. As an example, *Message Logger* shows how added value is generated from the APIs. *AMQP Math Tool Connector* and *AmqpRequestResponseHelper* provide an API solely for AMQP, but they can co-exist with *MessageSerialiser* as needed for data processing. The web services *Process Data Simulator* and *Requested History Data Publisher* communicate via AMQP using the message structures of COCOP information model.

3.6 Deployment and Use

The deployment and use of the toolkit is straightforward. The software components are distributed as Dynamic Link Libraries (.NET) or Java Archives, which are the native component formats of the platforms. The Message Logger is a desktop application and only requires .NET platform to run. The web services can be installed on any computer that runs NodeJs.

Listing 1 is an example about (1) serialising a message and (2) deserialising another in an object in C# language. The example is minimalistic but shows how straightforward the MessageSerialiser API is. After serialisation, the XML data would travel along the AMQP message bus. More thorough code examples are available in each API documentation [1].

[1] `https://kannisto.github.io/Cocop-Toolkit/`

Table 1: The items of COCOP Toolkit.

Item	Platform(s)	Description
– – **APIs (software components)**		
Cocop.MessageSerialiser.Meas	.NET, Java	APIs to serialise generic data as XML
Cocop.MessageSerialiser.Biz	.NET, Java	APIs to serialise production schedules as XML
Cocop.AmqpRequestResponseHelper	.NET, Java	APIs to implement request-response communication via AMQP
AMQP Math Tool Connector	Java	API to connect Matlab® with AMQP
– – **Web services**		
Process Data Simulator	NodeJs	Pushes historical process data to AMQP for simulation purposes; operated with Rest API
Requested History Data Publisher	NodeJs	Serves historical data over AMQP; request-response API over AMQP
– – **Desktop applications**		
Message Logger	.NET	Application to store and display messages from AMQP

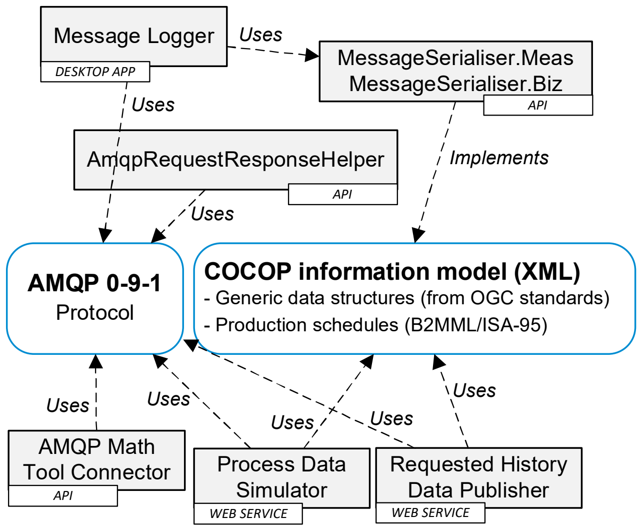

Fig. 2: The items in COCOP Toolkit and their relationship with the message bus.

Listing 1: Serialising a measurement value with Cocop.MessageSerialiser.Meas API (C#.NET version).

```
// 1. Create measurement message with metadata
var reading = new Item_Measurement("Cel", 22.7);
var obs = new Observation(reading) {
  FeatureOfInterest = "Plant2/Area200/TI203",
  ResultQuality = DataQuality.CreateGood()
};

// Serialize the message to publish to AMQP bus
byte[] xmlBytes = obs.ToXmlBytes();

// ...
// 2. Deserialize received message and
// get sensor reading
byte[] xmlBytesIn; // Raw data recvd from AMQP
var observationIn = new Observation(xmlBytesIn);
double ti203Reading = ((Item_Measurement)
     observationIn.Result).Value;
```

4. PRACTICAL INDUSTRIAL APPLICATIONS

COCOP Toolkit was utilised in communications in three prototypes. The domains include copper smelting, steel refinement and environmental assessment.

4.1 Plant-wide Coordination in Copper Smelter

The first use case is the plant-wide coordination of a copper smelter. The example smelter produces copper from sulfide ores with unit processes, such as *flash smelting*, *converting* and *anode refining and casting*. To operate efficiently, the processes must be coordinated, as each has a stateful nature and must control factors, such as the heat balance and composition of the material. The operation necessitates human expertise, but computerised advisory tools help, as the observation of processes is difficult due to heat, dirt and toxic substances. Besides, plant-wide coordination is a complex optimisation problem (Korpi et al., 2019).

The prototype is a mixture of publish-subscribe communication and legacy or proprietary technologies. The users are operators that receive advisory from user interfaces executed in Outotec ACT (Advanced Control Tool). This includes advisory for individual unit processes as well as a plant-wide scheduler. Furthermore, ACT manages the execution of tools that estimate the state of each unit process. For state estimation, all process data is retrieved over Open Platform Communications Data Access (OPC DA). However, state estimation also uses laboratory results and crane data retrieved from Structured Query Language (SQL) databases and communicated over the message bus whenever new information appears. The actual plant-wide coordination occurs in a scheduling software that instructs operators about the timing and parametrisation of the unit processes to reach what is considered the global optimum. The scheduler was developed with Matlab®. For illustration about the prototype, see Fig. 3 (a). In the message bus, all communication occurs with the publish-subscribe pattern. In the future, at least OPC DA communication should occur via the message bus instead.

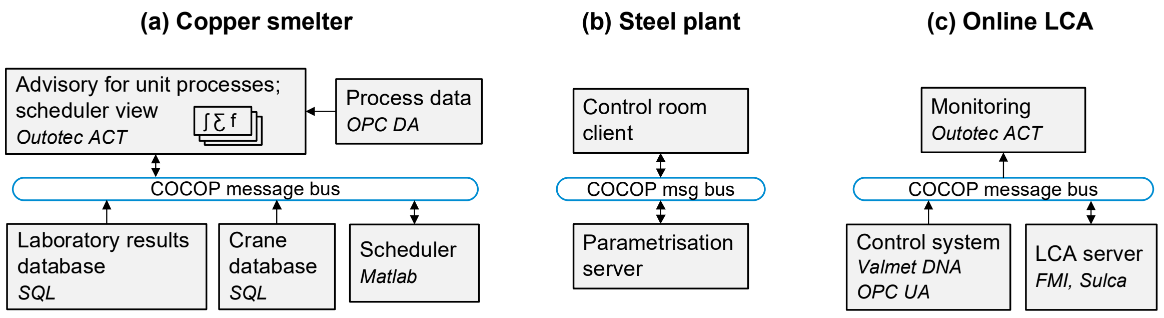

Fig. 3: Prototypes illustrated.

Because existing systems lack a compatible interface, adapters were developed. Matlab uses AMQP Math Tool Connector, whereas all other adapters were implemented in .NET with the APIs *Cocop.MessageSerialiser.Meas* and *Cocop.MessageSerialiser.Biz*.

4.2 Parametrisation in Steel Production

The second use case is to reduce the occurrence of surface defects in micro-alloyed steel products. In steel production, COCOP Toolkit was utilised in a prototype made for the parametrisation of the unit process *continuous casting*. To parametrise this process appropriately with steel temperature and thickness, related knowledge is necessary. This knowledge can be modelled in an advisory tool to help the operators.

The steel prototype receives advantage from the support for multi-paradigm communications in COCOP Toolkit. The user interface of operators is a client, and the actual parametrisation task occurs in a server. The task can take an arbitrary time to finish, so pure request-response would appear unresponsive to the user. Instead, an immediate response is returned to acknowledge the task, and the actual result is delivered later with the publish-subscribe pattern. For a better user experience, the task could return intermediate information about progress, but this was left as a future task. Furthermore, the prototype omits the connection with an actual knowledge module but still indicates the suitability of COCOP Toolkit for the application. For illustration, see Fig. 3 (b). The COCOP Toolkit APIs utilised are *Cocop.AmqpRequestResponseHelper* and *Cocop.MessageSerialiser.Meas* implemented in .NET.

4.3 Online Life Cycle Assessment

The third use case implements Online Life Cycle Assessment (Online LCA) to estimate the environmental impacts of production processes during operation. This approach enables LCA to guide plant operators in an active manner rather than restrict to offline estimates. In COCOP, Online LCA was implemented in a distillation column of industrial ethanol in a laboratory. The column runs actual industrial equipment, such as pumps and valves, and a *Valmet DNA* control system. To perform experiments, an LCA server was developed to estimate the realised contribution of the process to climate change (carbon dioxide equivalent) and acidification (sulfur dioxide equivalent).

The developed prototype consists of three components that communicate via COCOP message bus (see Fig. 3 (c)). First, LCA results are visualised in Outotec ACT that has been connected to the message bus via an adapter. ACT also shows actual measurement values from the distillation process. Second, the measurement values come to the message bus from Valmet DNA via another adapter that retrieves them from an OPC UA interface. Third, the actual LCA occurs in an Online LCA server that receives measurement values from the message bus and publishes the respective LCA results. The LCA model has been exported from a tool called *Sulca* and has a Functional Mockup Interface (FMI) connected to the message bus with an adapter. All adapters were implemented in .NET with *Cocop.MessageSerialiser.Meas* API. The architecture was earlier presented by Kannisto et al. (2019).

5. DISCUSSION AND CONCLUSIONS

This paper presented *COCOP Toolkit* to implement industrial communications in a loosely coupled, interoperable manner. The toolkit helps in implementing the COCOP architecture, which is based on AMQP 0-9-1 protocol and a common information model (Hästbacka et al., 2018a). These enable interoperability between software platforms in the plant-wide scope as well as data-driven applications, both of which gain importance in future industrial systems (Panetto et al., 2019).

COCOP Toolkit comprises various software items that aid the developers of software and simulation tools. An API facilitates the implementation of request-response communication, whereas another provides an API for message serialisation. A third API enables connectivity with a mathematics environment (Matlab$^{®}$). Furthermore, web services feed historical data to enable the testing of stateful simulation models. This enables the unit testing of optimisation modules with real data, which an important but not always easy task. However, the unit testing can reduce development time and costs, as it enables early corrective actions. Finally, a logging application facilitates error detection, which helps not only in development but also in commissioning if problems occur.

The control software presented in Section 4 demonstrated the applicability of the toolkit. A total of three solutions was introduced, each in a different production system: copper smelter, steel plant and distillation column for ethanol. In each solution, the toolkit reduced development effort in communications, so the developers could concentrate on application logic instead.

COCOP Toolkit is analogous with the OPC UA technology family, as this aims at interoperability in multi-vendor environments. However, COCOP Toolkit has features excluded from OPC UA, as this currently lacks publish-subscribe communication with message routing supported (OPC UA PubSub, 2017). Therefore, the future OPC UA should address this issue, as this would enable more scalable communications in the plant-wide scope.

There are still tasks of future research. COCOP Toolkit could be experimented in additional industrial use cases. Especially there could be studies in the manufacturing of discrete products or assembling machines, as the previous studies have observed only processes industry. Furthermore, the toolkit could be aligned with OPC UA and possibly aim at compatibility, although COCOP Architecture has features excluded from OPC UA.

ACKNOWLEDGEMENTS

The following persons in particular contributed to the development of the toolkit: Jussi Leinonen and Jukka Jääskeläinen from Metso Outotec, Finland; Olli Suominen and Tapio Vaaranmaa from Tampere University, Finland; Kari Rainio from VTT Technical Research Centre of Finland; Peter Craamer and Zigor Lopez from Mondragon Sistemas, Spain.

REFERENCES

AMQP 0-9-1 (2008). AMQP: Advanced message queueing protocol version 0-9-1. URL `http://www.amqp.org/specification/0-9-1/amqp-org-download` [Retr. 27 Aug 2020].

B2MML (2013). Business to manufacturing markup language. URL `http://www.mesa.org/en/B2MML.asp` [Retr. 27 Aug 2020].

Boyd, A., Noller, D., Peters, P., Salkeld, D., Thomasma, T., Gifford, C., Pike, S., and Smith, A. (2008). SOA in manufacturing guidebook. MESA International. URL `ftp://public.dhe.ibm.com/software/plm/pdif/MESA_SOAinManufacturingGuidebook.pdf` [Retr. 11 May 2020].

Cheng, Y., Chen, K., Sun, H., Zhang, Y., and Tao, F. (2018). Data and knowledge mining with big data towards smart production. *J. Ind. Inf. Integr.*, 9, 1–13. doi:10.1016/j.jii.2017.08.001.

COCOP (2020). COCOP SPIRE H2020 project. URL `https://cocop-spire.eu/` [Retr. 14 Aug 2020].

de Souza, L.M.S., Spiess, P., Guinard, D., Köhler, M., Karnouskos, S., and Savio, D. (2008). SOCRADES: A web service based shop floor integration infrastructure. In *The Internet of Things: First International Conference IOT 2008*, 50–67. Springer. doi:10.1007/978-3-540-78731-0_4.

Goldin, E., Feldman, D., Georgoulas, G., Castano, M., and Nikolakopoulos, G. (2017). Cloud computing for big data analytics in the process control industry. In *25th Mediterranean Conference on Control and Automation (MED)*, 1373–1378. doi:10.1109/MED.2017.7984310.

Gosewehr, F., Wermann, J., Borsych, W., and Colombo, A.W. (2017). Specification and design of an industrial manufacturing middleware. In *2017 IEEE 15th International Conference on Industrial Informatics (INDIN)*, 1160–1166. doi:10.1109/INDIN.2017.8104937.

Hästbacka, D., Kannisto, P., and Vilkko, M. (2018a). Data-driven and event-driven integration architecture for plant-wide industrial process monitoring and control. In *IECON 2018 - 44th Annual Conference of the IEEE Industrial Electronics Society*, 2979–2985. doi:10.1109/IECON.2018.8591323.

Hästbacka, D., Kannisto, P., and Vilkko, M. (2018b). Information models and information exchange in plant-wide monitoring and control of industrial processes. In *10th International Joint Conference on Knowledge Discovery, Knowledge Engineering and Knowledge Management - Volume 3: KMIS*, 216–222. doi:10.5220/0006960602160222.

Kannisto, P. and Hästbacka, D. (2018). Asynchronous communication platform concept to coordinate large-scale industrial processes. *IFAC-PapersOnLine*, 51(11), 1403 – 1408. doi:10.1016/j.ifacol.2018.08.325. 16th IFAC Symposium on Information Control Problems in Manufacturing INCOM 2018.

Kannisto, P., Hästbacka, D., Rainio, K., Leinonen, J., Alarotu, M., Pajula, T., Savolainen, J., and Vilkko, M. (2019). Plant-wide communication architecture enabling online life cycle assessment. In *Automaatiopäivät 23*. URL `https://pdfs.semanticscholar.org/b850/3a07fd0cee2bda9cb2eb8b61ba07e57df04e.pdf` [Retr. 28 Aug 2020].

Karnouskos, S., Colombo, A.W., Bangemann, T., Manninen, K., Camp, R., Tilly, M., Sikora, M., Jammes, F., Delsing, J., Eliasson, J., Nappey, P., Hu, J., and Graf, M. (2014). The IMC-AESOP architecture for cloud-based industrial cyber-physical systems. In A.W. Colombo et al. (eds.), *Industrial Cloud-based Cyber-physical Systems: The IMC-AESOP Approach*, 49–88. Springer. doi:10.1007/978-3-319-05624-1_3.

Korpi, M., Suominen, O., Jansson, J., Pihlasalo, J., and Vilkko, M. (2019). Plant-wide optimization of a copper smelter: How to do it in practice? In *European Metallurgical Conference EMC 2019 Volume 1*, 95–106.

OGC (2020). Open geospatial consortium. URL `https://www.ogc.org/` [Retr. 27 Aug 2020].

O'Hara, J. (2007). Toward a commodity enterprise middleware. *Queue*, 5(4), 48–55. doi:10.1145/1255421.1255424.

OPC Products (2020). URL `https://opcfoundation.org/products` [Retr. 27 Aug 2020].

OPC UA (2017). OPC unified architecture specification part 1, overview and concepts, 1.04. OPC Foundation.

OPC UA PubSub (2017). OPC unified architecture specification part 14, PubSub, release 1.04.

Panetto, H., Iung, B., Ivanov, D., Weichhart, G., and Wang, X. (2019). Challenges for the cyber-physical manufacturing enterprises of the future. *Annu. Rev. Control*, 47, 200–213. doi:10.1016/j.arcontrol.2019.02.002.

Sarnovsky, M., Bednar, P., and Smatana, M. (2018). Big data processing and analytics platform architecture for process industry factories. *Big Data Cogn. Comput.*, 2(1), 3. doi:10.3390/bdcc2010003.

Theorin, A., Bengtsson, K., Provost, J., Lieder, M., Johnsson, C., Lundholm, T., and Lennartson, B. (2017). An event-driven manufacturing information system architecture for Industry 4.0. *Int. J. Prod. Res.*, 55(5), 1297–1311. doi:10.1080/00207543.2016.1201604.