Organizing IoT Systems-of-Systems from Standardized Engineering Data

Oscar Carlsson
Midroc Automation AB
Stockholm, Sweden
Email: oscar.carlsson@midroc.se

Csaba Hegedűs
AITIA International, Inc.
Budapest, Hungary
Email: hegeduscs@aitia.ai

Jerker Delsing
Luleå University of Technology
Luleå, Sweden
Email: jerker.delsing@ltu.se

Pal Varga
Budapest University of Technology and Economics
Budapest, Hungary
Email: pvarga@tmit.bme.hu

Abstract—Tackling current challenges in production automation requires the involvement of new concepts like Internet of Things, System-of-Systems and local automation clouds. The objective of this paper is to address the actual process of defining a cloud based automation system. More specifically the design, engineering, operation and maintenance of an automation system must be captured and managed between all stakeholders involved. This is critical to create the expected benefits from the local automation cloud approach.

This paper addresses the capability of capturing plant designs and coordinating information exchange based on the captured architecture. For this purpose an architectural component – Plant Description – is proposed to be used in the Arrowhead Framework, based on already existing plant automation standards. An overview of methodologies on how it can interact with the Arrowhead Framework’s Orchestration process describes the usefulness in managing large-scale automation systems. A qualitative evaluation for one of the proposed approaches is also described in a water control use case that can be found both in process and building automation.

I. INTRODUCTION

Many of the current societal challenges revolve around efforts for sustainability, flexibility, efficiency and competitiveness. This trend, among others, is imposing new requirements on the technologies used to support production. Here a number of gaps have to be addressed regarding e.g. technologies, related business models and operational management.

Despite all the new operational and organisational ideas, the automation fundamentals captured by today’s state-of-the-art automation technologies have to be maintained. Thus the next generation of automation and digitalisation technology has to meet a large set of requirements and involving a wider scope of actors and stakeholders.

In 2011, the concept of Industry 4.0 [1] was born in Germany. This concept builds upon the last generation of industrial monitoring and control systems, but enables an even finer level of interaction between shop-floor devices and high-level enterprise systems. In Industry 4.0, Internet of Things (IoT), Cyber-physical Systems (CPS) and other concepts are to be utilized in order to break down the classical strict

hierarchical approach of ISA-95 [2], replacing it with a more flexible approach without barriers and closed systems.

For the transfer of the ISA-95 architecture to a cloud based approach much important work is already published, including: system architectures [3], [4]; suitable technologies [5]; real time considerations [6], [7]; migration from legacy systems [8], [9] and engineering for cloud based automation [10], [11]. This cross-compliance is depicted on Fig. 1 based on approaches of the SOCRADES [12] and IMC-AESOP [13] projects. Moreover, a parallel discussion is also present for the MES and ERP levels. Some important works on cloud approaches are [3] and [14].

The concept of local automation clouds have recently been introduced via the Arrowhead project through the newly released Arrowhead Framework [15] and the book edited by Delsing [16]. Automation clouds have a number of important properties to consider. One group of its properties is based on how a “production plant” is described and viewed. Such a plant descriptor tool should be capable of addressing a number of application requirements and therefore should reflect the engineering, deployment, operation and maintenance features of the “plant”. The following list enumerates a couple of usage scenarios:

- 1) Plant design and System of Systems interaction design at design time
- 2) Management of run-time changes made in the plant, i.e. replacement of devices or rearrangement of the System of Systems interaction
- 3) Supporting automated deployment procedures, plant commissioning, plant restart and plant updates
- 4) Run-time creation of System-of-Systems orchestration rules based on physical processes
- 5) Extraction and monitoring of current plant status
- 6) Supporting run-time error and fault handling processes, possibly with alternative orchestration configurations
- 7) Optimizing production and physical processes

The objective of this paper is to primarily address the forth and fifth items of this list. Thus the detailed objectives are to

Fig. 1. Transferring the ISA-95 automation pyramid architecture to a local automation cloud architecture.

explain how to capture “plant” changes at run-time and distil updated System-of-Systems (SoS) orchestration rules based on them. It is also of interest to recognize “plant” operation errors and faults requiring reconfiguration of the SoS within the Arrowhead framework.

To meet these, section II outlines the related work, followed by section III, which presents the proposed approach and the integration within the Arrowhead framework. Section IV describes a practical use case, and then Section V discusses the conclusions and maps out the future work.

II. RELATED WORK

A. Process Engineering State of the Art

There are already several standards and groups of standards that allow the integration of engineering data from different sources. Although they use different approaches, most of them appear to aim for full integration of all data into one large database. Yang describes many of the challenges and strategies that has fostered this approach [17].

One example is the group of standards put forward by the ISO Technical Committee 184, Subcommittee 4 (ISO/TC 184/SC 4)¹. They cover many different aspects of industrial information and data, and also include the ISO 18876 that presents a methodology specifically for integrating industrial data based on different data models into one extended data model.

Another such group of standards is based around IEC 62714 (AutomationML) [18], [19]². AutomationML uses a tree-structure that is used to organize all objects to describe the plant topology. This gives the defined hierarchy higher priority than any other object relations that can be represented by

using separate link-objects called CAEX-InternalLinks [20]. Lüder et al. [21] have shown how additional information (not specified in the standard) can be appended to an existing AutomationML model, allowing all engineering data to be contained in one consistent data set.

The former group of standards appears to be more popular in the process industries and the American defense industry, while the latter appears to be more popular in manufacturing industries. Holm et al. [22] presents a comparison of the plant modeling concepts described in ISO 15926 and IEC 62424, respectively.

B. The Arrowhead Framework

The Arrowhead framework uses a service-oriented approach to tackle interoperability and integrability issues within closed or separated automation environments called *local clouds*. It does so by facilitating and governing the service interactions between *Application Systems* using the *Arrowhead Core Systems*, see Figure 2 [16].

Fig. 2. Arrowhead Core Systems and their use

A *service* in the Arrowhead Framework is what is used to exchange information from a providing System to a consuming

¹This group includes the ISO 8000, ISO 10303, ISO 13584, ISO 15926 and ISO 18828 standards.

²Such as IEC 62424 (CAEX), ISO/PAS 17506 (COLLADA) and PLCopen.

System, e.g. a sensor readout or a command for an actuator. The Core Systems of the Arrowhead Framework help to establish and manage the service connections, but the service interaction is performed directly between the application systems. There are three mandatory Core Systems that are required to form a local cloud, as depicted in Figure 2 – however additional core systems can be used to support different aspects of a local cloud, the Plant Description System being one of them.

The Service Registry stores the service offerings of each registered system. As its name suggests, the Authorization System manages AA tasks. Meanwhile, the Orchestrator is responsible for instrumenting each System in the cloud: where to connect and what to consume. It instructs so by pointing towards specific Service Producers to consume specific Service(s) from (giving hence an *orchestration rule*). This orchestration service can, however, be provided through various internal methods (tailored to use cases), as discussed in section III-B. The orchestration rule itself merely has to point to a Service Producer and declare the Service that is to be consumed at minimum. If required, further information can and should also be passed on (i.e. authorization information, service metadata or validity period).

1) *The Plant Description*: A “plant” in this context within the Arrowhead framework refers to a larger group of systems acting together, often defined by a geographical or organizational boundary. A typical plant would be a factory, a mine, or a water treatment plant, but it could as well be a hospital, a windmill farm or an airport. For many purposes it is necessary to group systems together into hierarchies, and describe how the systems relate to each other. There are already numerous standards exist to describe these hierarchies, but since many disciplines are required to build a plant, there are often many different hierarchies from different standards within one plant.

Fig. 3. Example of how objects may be identified by more than one viewpoint, based on IEC 81346 [23]. The green objects in the middle are groups of electrical equipment that are both located together and perform a function together.

Figure 3 illustrates how a device or component – documented according to IEC 81346 [23] – may be found through traversing two different hierarchies, depending on the interest of the user. In this figure the green objects in the middle are groups of electrical equipment (switches, contactors and fuses) that are both placed at the same location, and perform a function together. The role of this group of objects – as per

the function they provide – can be derived by traversing the red hierarchy on the right of the figure, while their physical location can be found by traversing the blue hierarchy on the left. Each colour on the figure can be represented as a separate viewpoint by the Plant Description, i.e. blue representing the location of the object or green representing the components making up a product.

The objective of the Plant Description therefore has been primarily to provide a basic common understanding of the layout of a “plant”, providing possibilities for actors with different interests and viewpoints to access their view of the same data set. Details of the design and prototype implementation of a Plant Description are described by Carlsson et al. [24]. For its use as a source of design data for System-of-Systems organization, its most important features are (i) the simplified description of object relations, (ii) topologies, and (iii) a unified data storage node structure.

III. APPROACH

The creation of orchestration rules from relations between automation objects – as set by design and engineering tools – should be automated. Defining such an automated process would simplify deployment and commissioning procedures, as well as enable run-time creation of orchestration rules. If set-up correctly it could also allow for creation of alternative orchestrations, that could be used to mitigate certain run-time errors and automate fault handling. This section is intended to present how the Plant Description could be used to automate the creation of orchestration rules.

A. Plant Description

The Plant Description is used for identifying the objects and their relations while leaving the object details in the already established, standardized databases provided by existing engineering tools. The objects are stored as nodes, containing only the identifiers used in the respective engineering databases. The object relations are stored as separate links between nodes, with an attribute to each link denoting what kind of relationship the link represents.

Figure 4 shows a simplified illustration of how the Plant Description can be used to construct an orchestration rule. To the left, in blue, are the nodes and links in the Plant Description describing automation objects and their relations. One of these links, called the Functional service link, describes that the automation object at the top is to have a service interaction with the automation object at the bottom. In other words there is a Functional relationship between those nodes at the top and bottom – and it is a service interaction. It is important to note that the automation objects in the Plant Description are merely logical representations and not the actual hardware and software that performs the automation. To the right, in green, in figure 4 are the actual Systems (i.e. a software running on some hardware in the physical plant). In order to create an orchestration rule between the top System and the bottom System, it is enough to have knowledge of the links and nodes in the Plant Description and the knowledge of

which System represents which node. If both of these pieces of data are available to an “Orchestrator”, such rules can be created automatically.

Fig. 4. Illustration of how a relationship in the Plant Description can translate into an orchestration rule.

An extension of the Plant Description that is only conceptual so far, is to allow a certain degree of flexibility in the connections between objects. This concept states that the Plant Description should describe alternative service links, so that some but not all of the designed service links can appear as orchestration rules. This can be done in more than one way but it requires either an additional parameter for each link, or that there should be different link-types for the different kinds of service links. For flexible orchestrations they could be described in the Plant Description either as multiple link-objects – each link connecting one source node to one target node –, or as single links containing multiple sources and/or targets, where not all connections should be realised as orchestration rules.

B. Orchestration of systems

Orchestration in the Arrowhead context refers to a supportive task, namely providing instructions to Application Systems on their service consumption behaviour, as discussed in section II-B. Since this is merely a black-box definition for the Orchestration service (describing its inputs, outputs, and interfaces), it provides room for various technological approaches on its internal logic and implementation. Therefore, a high-level design decision can be made on how to build the orchestration process that best matches service-consuming Systems to producing Systems. This degree of freedom can result in fundamentally different “Orchestrators” that fit the different needs of various use cases. Currently, there are three archetypes of Orchestrators in scope within Arrowhead. This

section explores these and how they can be configured to operate in a “plant”.

1) *Static rule-sets*: In certain cases, static rule-sets are sufficient to orchestrate complex Systems-of-Systems since their compositions do not change over time. In this case, every system in the local cloud has a set of partners pre-defined and they cannot connect anywhere else. If this composition changes, however, it can only be carried out through human intervention in a process re-engineering procedure. The description of the System-of-Systems only changes during these reconstructions or tweaks. Here, the orchestration rule-set is also manually created.

For these cases, the Plant Description is rather a process engineering tool that supports these transitions. It describes how the SoS is laid out and realized with physical devices. This way, devices then can be mapped into systems in the Arrowhead aspect, and have their orchestration data re-entered in the orchestration store. Therefore, being user-friendly is a major requirement towards the plant description in this scenario.

As an example for this use-case, let us consider an actual production plant where Arrowhead services are physical processes and Arrowhead systems are e.g. actuators, sensors or controllers. For such cases, certain sensors can only provide data for specific controllers and the same goes for actuators. These static restrictions on connections can be translated into orchestration rules. The task of the orchestrator is merely to support the (re-)establishment of these connections. Nevertheless, an Orchestrator using static rule-sets is still beneficial, since proper Plant Description tools can make it easy to reprogram a production line.

2) *Dynamic orchestration*: For those use-cases that are challenged by changing environments and system set-up, the Orchestrator system needs to become a dynamic matchmaker that brings together service consumers with providers at run-time. However, a such matchmaker should naturally not operate in a free-for-all manner. Even though Systems might be able to connect to every other System, such ad hoc connections might still not be desired (or e.g. authorized). Therefore, dynamic service discovery has to be augmented with admission control and limited by other factors – such as physical or (real-time) resource-related restrictions in the local cloud as depicted on Figure 5.

For these cases, the Plant Description functionality could provide general configuration information, as well as information of alternative rule-sets (e.g. describing alternative connections), and the preferences of their usage. Utilizing the functionality of different viewpoints in the Plant Description, the different Core Systems would be able to access the objects and relationships relevant to their functionality. This approach further eases the configuration for the Core Systems, as it reduces the risk of inconsistency in data used by different Core Systems. However, it does add further complexity to the input and management of Plant Description data.

3) *Automated process engineering*: In previous cases the orchestration process worked in a solicited manner: requested

Fig. 5. A Plant Description Engine providing configuration for the Core Systems

on pre-set criteria (e.g. events) or by service consumers. A third option is automating the process engineering, enabling unsolicited orchestration. The matchmaking of service consumers and providers is further optimized here by a response on the status of the SoS – based on measured operational variables. In this case the trigger of orchestration is not a direct request from any system; it rather comes from an inferred decision; based on the up-to-date knowledge about the SoS. The concept is very similar to the initial idea of Cognitive Networks, Software Defined Networking, or the Knowledge Plane approach [25].

The overall configuration of the System-of-System will be optimized for the global target of the given local cloud (e.g. setting a production target for production lines, or optimal energy consumption for a smart building). This sort of Orchestrator is then able to configure all systems to reach the higher goal: what services they need to consume, from where, or for how long. Such an Orchestrator engine would naturally have to utilize Plant Description related data and its operation must rely on the functional, (physical) architectural and process-oriented mappings of the SoS.

In such scenarios, the decisions and acts of such an engine is asynchronous to the general operations within a local cloud. It is only invoked to change the general flow in order to enforce the existing operational targets on the SoS (or push new ones). This means that the Plant Description in this case will be used to store required, desired or recommended connections between design objects.

IV. USE CASE: ORCHESTRATION FROM PROCESS & INSTRUMENTATION DIAGRAM

Pang et al. [26] have shown how a system described by an IEC 62424 Process and Instrumentation Diagram (P&ID) can be formally transformed to IEC 61499 applications. Using part of the P&ID in their example to create a Plant Description, it can then be used to illustrate how orchestration rules might be generated for a System-of-System, based on service interactions.

Fig. 6. P&ID of a water control loop, including a Flowmeter (FI), a Thermometer (TI), a Control valve (YC), a Controller (UC) and a User interface (HIC); all used to control the flow through the Valve V-0-1 into the Tank B-0-1.

Figure 6 shows the portion of the P&ID to be used in this example. As an implementation using the Arrowhead framework, this can be described as five systems – all belonging to the function “Water control loop” –, and between these systems there are four service interactions. In the Plant Description, these five systems would be considered Nodes, given a unique NodeId and an identifier within the CAEX-structure that would allow a user to access further information through there. Each link between these systems would be given a few parameters: (i) unique LinkId, identifiers for the link’s (ii) source and (iii) target nodes, and (iv) link type. An example of what the data in a Plant description might look like is given in the left part of Figure 7.

Beside giving this representation of the systems and their relations, the Orchestration engine also needs a connection between the nodes in the Plant Description and the running systems requesting orchestration rules. This information could be stored either in the Plant Description, as an additional parameter for each node – by the system itself if the NodeId is given to the system as part of the configuration process –, or in a separate System Registry where additional information about each system is stored as a central repository.

As the Orchestration engine has access to the two pieces of information; i) which system instantiates each node and ii) which nodes should be connected by service interactions, it can compose orchestration rules either on request as systems are connected or as a result of new links being created in the Plant Description as a result of an updated design.

Fig. 7. An example for the control elements and their relations, as represented in the Plant Description, and how this information can be used to create orchestration rules. The blue parts illustrate objects in the Plant Description, the green are Systems that can be orchestrated and the yellow are service production and consumption interfaces that are to be connected through service interactions.

V. CONCLUSIONS AND FUTURE WORK

To address a dynamic and changing production environment, it will be important that the configuration of large systems can be changed easily, in a structured manner, according to the intentions of the engineer or designer that initiated the change. The approach presented here shows how the dynamic nature of an automation system based on service interactions can be orchestrated, utilizing standardised engineering data, provided by existing engineering tools.

Additional useful information for the orchestration engine includes the current state of the plant, and how prioritisation can be made based on the current plant state and the current production recipe. Thus future work has to address ways of detecting plant state in run-time, and means of providing service quality information suitable for dynamic orchestration enabling various types of optimisation.

ACKNOWLEDGMENT

The authors would like to extend their gratitude towards EU ARTEMIS JU for funding within project ARTEMIS/0001/2012, JU grant nr. 332987 (ARROWHEAD). We would also like to thank the partners of the Arrowhead project for fruitful collaboration and interesting discussions.

REFERENCES

- [1] J. Lee, B. Bagheri, and H.-A. Kao, "A cyber-physical systems architecture for industry 4.0-based manufacturing systems," *Manufacturing Letters*, vol. 3, pp. 18–23, 2015.
- [2] B. Scholten, *The Road to Integration: A Guide to Applying the ISA-95 Standard in Manufacturing*. ISA, 2007.
- [3] S. Karnouskos and A. W. Colombo, "Architecting the next generation of service-based scada/dcs system of systems," in *Proceedings IECON 2011*, Melbourne, Nov. 2011, p. 6.
- [4] F. Jammes and H. Smit, "Service-oriented paradigms in industrial automation," *Industrial Informatics, IEEE Transactions on*, vol. 1, no. 1, pp. 62–70, Feb 2005.
- [5] F. Jammes, B. Bony, P. Nappey, A. W. Colombo, J. Delsing, J. Eliasson, R. Kyusakov, S. Karnouskos, P. Stluka, and M. Till, "Technologies for soa-based distributed large scale process monitoring and control systems," in *IECON 2012-38th Annual Conference on IEEE Industrial Electronics Society*. IEEE, 2012, pp. 5799–5804.
- [6] G. Candido, A. W. Colombo, J. Barata, and F. Jammes, "Service-oriented infrastructure to support the deployment of evolvable production systems," *IEEE Transactions on Industrial Informatics*, vol. 7, no. 4, pp. 759–767, Nov 2011.
- [7] R. Kyusakov, P. P. Pereira, J. Eliasson, and J. Delsing, "Expip: a framework for embedded web development," *ACM Transactions on the Web (TWEB)*, vol. 8, no. 4, p. 23, 2014.
- [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 on IEEE Industrial Electronics Society*, Nov 2011, pp. 4472–4477.
- [9] J. Delsing, F. Rosenqvist, O. Carlsson, A. W. Colombo, and T. Bange-mann, "Migration of industrial process control systems into service oriented architecture," in *IECON 2012*. IEEE, 2012.
- [10] A. Jain, D. Vera, and R. Harrison, "Virtual commissioning of modular automation systems," in *Intelligent Manufacturing Systems*, vol. 10, no. 1. IFAC, 2010, pp. 72–77.
- [11] N. Kaur, C. S. McLeod, A. Jain, R. Harrison, B. Ahmad, A. W. Colombo, and J. Delsing, "Design and simulation of a soa-based system of systems for automation in the residential sector," in *IEEE ICIT 2013*. IEEE, 2013.
- [12] A. W. Colombo, ed. (2016) Socrates project. [Online]. Available: <http://www.socrates.net>
- [13] A. W. Colombo, T. Bangemann, S. Karnouskos, J. Delsing, P. Stluka, R. Harrison, F. Jammes, and J. L. Lastra, "Industrial cloud-based cyber-physical systems," *The IMC-AESOP Approach*, 2014.
- [14] S. Karnouskos, A. W. Colombo, F. Jammes, J. Delsing, and T. Bange-mann, "Towards an architecture for service-oriented process monitoring and control," in *IECON 2010-36th Annual Conference on IEEE Industrial Electronics Society*. IEEE, 2010, pp. 1385–1391.
- [15] J. Delsing, ed. (2016) Arrowhead framework wiki. [Online]. Available: <http://forge.soa4d.org/arrowhead-f/wiki>
- [16] J. Delsing, ed., *IoT based Automation - made possible by Arrowhead Framework*. CRC Press, Taylor & Francis Group, 2016.
- [17] R. Yang, *Process Plant Lifecycle Information Management*. AuthorHouse, 2009. [Online]. Available: <https://books.google.se/books?id=zmgzdjflGR0C>
- [18] M. Schleipen, R. Drath, and O. Sauer, "The system-independent data exchange format caex for supporting an automatic configuration of a production monitoring and control system," in *2008 IEEE International Symposium on Industrial Electronics*, June 2008, pp. 1786–1791.
- [19] M. Schleipen and M. Okon, "The CAEX tool suite - User assistance for the use of standardized plant engineering data exchange," in *Emerging Technologies and Factory Automation (ETFA), 2010 IEEE Conference on*, Sept 2010, pp. 1–7.
- [20] R. Drath, A. Luder, J. Peschke, and L. Hundt, "AutomationML - the glue for seamless automation engineering," in *2008 IEEE International Conference on Emerging Technologies and Factory Automation*, Sept 2008, pp. 616–623.
- [21] A. Lüder, N. Schmidt, R. Rosendahl, and M. John, "Integrating different information types within automationml," in *Proceedings of the 2014 IEEE Emerging Technology and Factory Automation (ETFA)*, Sept 2014, pp. 1–5.
- [22] T. Holm, L. Christiansen, M. Gring, T. Jger, and A. Fay, "ISO 15926 vs. IEC 62424 - Comparison of plant structure modeling concepts," in *Proceedings of 2012 IEEE 17th International Conference on Emerging Technologies Factory Automation (ETFA 2012)*, Sept 2012, pp. 1–8.
- [23] *ISO/IEC 81346-1:2009, Industrial systems, installations and equipment and industrial products - Structuring principles and reference designations*, ISO/IEC Std.
- [24] O. Carlsson, D. Vera, J. Delsing, B. Ahmad, and R. Harrison, "Plant descriptions for engineering tool interoperability," 2016, submitted to INDIN 2016 IEEE International Conference on Industrial Informatics.
- [25] D. D. Clark, C. Partridge, and J. C. Ramming, "A knowledge plane for the Internet," in *SIGCOMM*, Aug 2003, pp. 3–10.
- [26] C. Pang, V. Vyatkin, and W. Dai, "Iec 61499 based model-driven process control engineering," in *Proceedings of the 2014 IEEE Emerging Technology and Factory Automation (ETFA)*, Sept 2014, pp. 1–8.