Configuration Service in Cloud based Automation Systems

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

Pablo Puñal Pereira*, Jens Eliasson† and Jerker Delsing‡
Luleå University of Technology
Luleå, Sweden
Email: pablo.punal@ltu.se*, jens.eliasson@ltu.se†, jerker.delsing@ltu.se‡

Bilal Ahmad* and Robert Harrison†
Automation Systems Group, WMG, the University of Warwick
Coventry, United Kingdom
Email: b.ahmad@warwick.ac.uk*, robert.harrison@warwick.ac.uk†

Ove Jansson
Abelko Innovation
Luleå, Sweden
Email: ove.jansson@abelko.se

***Abstract*—Current challenges in production automation requires the involvement of new technologies like IoT, SoS and local automation clouds. The objective of this paper is to address the actual process of defining a cloud based automation system. The objective of this paper is to address one of the challenges involved in establishing and managing a cloud based automation system. Three key capabilities have been identified as required to create the expected benefits of local automation clouds; 1) capturing of plant design 2) capturing and distributing configuration and deployment information 3) coordinating information exchange .**

This paper addresses the capturing and distribution of configuration and deployment information. For this purpose a SOA service is proposed, the ConfigurationStore, following the principles of the Arrowhead Framework. The service is accompanied by a deployment methodology and a bootstrapping procedure. These are discussed for several types of automation technology, e.g. PLC's, sensors, actuators. A qualitative evaluation of the proposed approach is made for four use cases; Building automation, Manufacturing automation, Process automation and IoT devices. Concluding the usability for large-scale deployment and configuration of Industrial Internet of Things.

I. Introduction

High level topics in today's society are sustainability, flexibility, efficiency and competitiveness. These in turn are driven by big societal questions like environmental sustainability, availability of energy and raw material, rapidly changing market trends. We find several trends that in different ways are addressing these topics. One is the move from large monolithic organisations towards multi-stakeholder cooperations where cooperation are fostered by market requirements. Another is the learning from previous products, other parts of the value chain, the life cycle of the product and the product or service production process itself.

These trends are creating new requirements for the technology used to support product and service production, causing a drive for digitisation of production. Digesting this reveals a number of gaps regarding technology, organisation, cooperation structure, operational management and related business models that have to addressed.

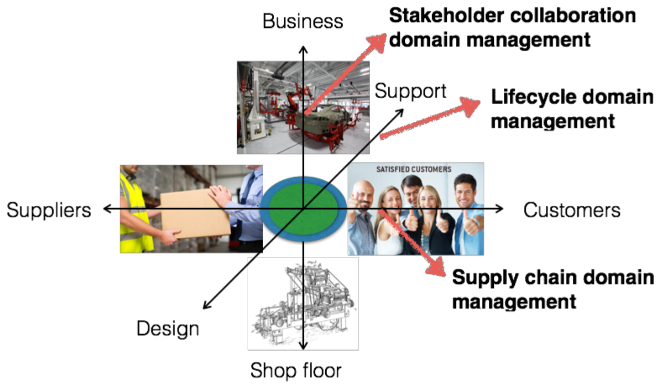
Business
Stakeholder collaboration domain management
Support
Lifecycle domain management
SATISFIED CUSTOMERS
Suppliers
Customers
Supply chain domain management
Design
Shop floor

Fig. 1. Three important axes for collaborative production, product life cycle, supply chain and stakeholder integration management

Around organisation, cooperation structure and operational management the high profile key aspects are related to three domains, see Figure 1:

- Product life cycle management
- Supply chain management
- Stakeholder integration management

With the move from large monolithic enterprises towards multi-stakeholder cooperation the management is changing towards distributed multi-stakeholder collaboration with distributed responsibilities and decision making. The flexible collaboration along and in-between the three domains also opens for the possibility of dynamic learning. A further aspect is that these domains tend to become wider (longer) involving more stakeholders with diverse objectives and more details and variations of the service or product to meet customer diversity and service and product quality.

These ideas are currently emerging but regarded as very important to address the high level topics of flexibility, efficiency, competitiveness and with suitable incentives also supporting sustainability. To support these developments there are a number of technology gaps which seemingly can not be addressed by current state of the art. Because of this, a number of new technologies are emerging to fill these gaps. Some current big buzz technologies are:

- Internet of Thing, IoT
- System of Systems, SoS
- Cyber-Physical Systems, CPS
- Cloud
- Big data
- Service Oriented Architecture, SOA

Despite all the new operational and organisational ideas and emerging technologies the automation fundamentals captured by today's state of the art automation technology have to maintained. Thus, next generation of automation and digitalisation technology has to meet a large set of requirements and involve a wider scope of actors and stakeholders. This is the big challenge for technology suppliers of the future, in this field.

ISA-95, standardised through ISA [1], is today's standard architecture for automations systems [2]. Accompanying the ISA-95 standard are related standards like ISA-99 and IEC 62443 [3], [4], which address control system security.

In 2011, the concept of Industry 4.0 [5] 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, state of the art technologies like Internet of Things (IoT) and Cyber-physical Systems (CPS) are utilized in order to be able to break the classical, strictly hierarchical, approach of ISA-95 [2] with a more flexible approach without fixed barriers and closed systems. By basing all communication on standards-based protocols, like the TCP/IP protocol suite, it is now possible to have information exchange between (almost) any systems in a production facility. This opens up for new strategies in terms of global plant optimization, minimized energy consumption, safety and security, etc.

The current limitations and bottlenecks are projected to be addressed by the introduction of concepts like Internet of Things (IoT), System of Systems (SoS) and various cloud technologies.

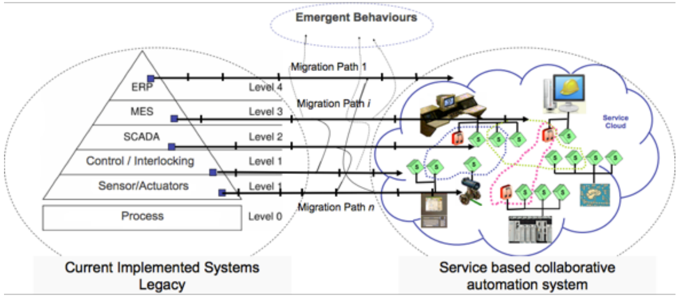

Fig. 2. Transferring the ISA-95 automation pyramid architecture to a cloud. This has been investigated by several larger EU projects like e.g. SOCRADES and IMC-AESOP.

The trends and perspectives put forward indicates that the current solutions use to build automation systems are not sufficient. Further the cost connected with the engineering and building of larger automation systems involving multiple stakeholders seems to be prohibitively high.

For the last 10 years, discussion on the next generation SCADA, DCS and MES systems has been around. A multitude of research projects on the topic have been executed. Some more prominent such are SOFIA, SOCRADES [6], and IMC-AESOP [7]. All of them were investigating a move from a hierarchical ISA-95 approach to a more cloud-like approach. A well-known illustration from the IMC-AESOP project is illustrating such move from the pyramid to the cloud in Figure 2. This is currently a rapidly changing landscape but some other efforts touching this field are e.g. FiWare [8]. In addition there are a growing number of cloud offerings on the market. An analysis of the different approaches found in 2015 can be found in [9].

In all these cases the key technology for creating the integration within and in-between different levels of the ISA-95 architecture is Service Oriented Architecture, SOA [10]. SOA was originally developed by IBM to enable data/information exchange between different lines of computer systems.

For the transfer of the ISA-95 architecture to a cloud based approach we do find some important published work regarding, system architecture [11], [12], suitable technologies [13], real time [14], [15] migration from legacy systems [16], [17], engineering for cloud based automation [18], [19].

A parallel discussion is ongoing for the MES and ERP levels. Some important publications on cloud approaches for the MES and ERP level are [11], [20].

Recently the concept of local automation clouds have been introduced via the Arrowhead project the newly released Arrowhead Framework [21].

Most of the recent developments are adopting Service Oriented Architecture, SOA, as the main approach to enable plant automation using cloud technology.

The objective of this paper is to take a step beyond the current cloud automation technology for production. The ambition is to address the actual process of producing a cloud based automation system. It is here argued that there are at least three important capabilities of the automation clouds which are critical to create expected benefits from local automation cloud approach. The capabilities are:

- A way of capturing a plant design: physical devices, components and systems, geographical layout and controlled interaction between physical devices, components and systems
- A way of capturing and distributing configuration information to the entities involved in the plant design
- A way of coordinating information exchange between different entities within a plant

This paper will address the second point: capturing and distribution of configuration information.

The outline is; Section II outlines related work, followed by Sections III and IV, which presents the proposed approach and gives examples of a few different use cases with experimental results. Section V gives a theorethical discussion on the findings, followed by the conclusions in Section VI. The paper finishes by stating ideas for future work in Section VII.

II. Related work

The configuration has long been a significant part of commissioning automated systems in all parts of society. As different domains and applications form, using different approaches and technology, the configuration procedure has developed in somewhat different directions in different areas.

Hodek and Schlick [22] describe the typical operations for integration of field devices in a state-of-the-art industrial automation system. The device profiles that they present could very well be used as a standardized baseline configuration for field devices. However, they also note that there are several other components to a quick and simple commissioning, such as the programming of Programmable Logic Controllers (PLC's) and integration into enterprise software systems. Additionally one of the suggested further investigations is on technology independent Plug& Play features of industrial Ethernet solutions.

Cachapa et al. [23] show how an engineering tool based on a Service-Oriented Architecture (SOA) can help by simplifying the process of designing and applying a production line layout, although in their case the configuration still contains manual configuration for each device.

Dürkop et al. [24] present a solution for a reconfigurable automation system where a Programmable Logic Controller (PLC), using Real-time Ethernet (RTE) connected IO-devices, is equipped with a Web-Service Interface that allows configuration of both the PLC and the RTE network.

Perera et al. [25] present a model to help users configure IoT middleware, primarily for middleware used to collect and process data from IoT-enabled sensors. The issue described here illustrates a situation that can be expected to become more common as automation systems become more dynamic and reconfigurable, where users or operators without detailed knowledge in IT or automation are tasked with the reconfiguration of a system and how tools and methods can aid this process.

III. Proposed Approach

As discussed in the previous sections, there are several approaches and solutions that can help improve the engineering and configuration of both existing and future automation systems in many different areas.

The approach presented in this paper attempts to define certain methods, structures and interaction patterns that will fit in the many areas of society that Arrowhead aims to address. As this will encompass a large number of different technologies and very different external requirements, the general approach is not detailed on a technical level but instead some more detailed scenarios are discussed in the following sections.

A. Arrowhead Configuration store

The Configuration Store service is one of the Arrowhead automation support core services.

The purpose of the Configuration Store is to provide a uniform way for Arrowhead compliant system to manage distribution of configurations. The extent of the configurability of a system ultimately depends on the system and therefore the design of this service is intended to allow different levels of configurations to be transferred using the same interface, from changes in system parameters to full firmware updates that may change which services a system is able to produce and consume.

Through this design the decision of how configurable a system should be is left to the system provider, and not imposed by the framework, while still allowing a uniform method for configuration management across diverse systems of systems containing systems with different levels of configurability.

In this design the actual configuration file is not necessarily provided by the Configuration Store, this design was selected to accommodate difference scenarios where accessibility, storage or file transfer ability may otherwise be limiting the distribution of configurations. There is however the possibility to have the same system providing both the Configuration Store service and the appointed file storage.

B. General deployment procedure

As the Arrowhead Framework is intended to allow cost effective deployment of a very wide array of devices, all general methods and procedures need to allow for some flexibility and adaptation to the specific use case. Still, many Arrowhead compatible devices are expected to be deployed in large numbers using low-cost hardware manufactured identically in very large numbers. Under these conditions it becomes even more important to keep the costs for engineering, deployment and commissioning at a minimum.

For the general procedure, the devices are assumed to have identical hardware and identical software preloaded from a factory, workshop or back office, the only differences between devices are their network Media Access Control address (MAC address) and their Serial Number (S/N), or some other individual identifier that has been assigned to them automatically during the device manufacturing process. The preloaded software contains the required security measures and to allow the device to connect to the Arrowhead Framework Core systems and to use a minimal set of services. The basic security may, for example, consist of a certificate installed as part of the manufacturing process which certifies that the device is of the brand and type that it claims to be.

To organize the large number of devices into a productive system of systems each device needs to be assigned a specific task and be configured to perform the task according to a larger plan. This information describing the different tasks and configurations is store in the Configuration Store, as described in section III-A. In order to allow some flexibility in network structure, without increasing the engineering or deployment time for each device, a specific procedure has been developed.

Step-by-step general deployment procedure:

1) Device is physically connected to the network and turned on.
2) Device connects to the network using DHCP, or a similar standardized technology applicable for the network in question.
3) Device looks for the Arrowhead core systems [26] at predefined locations. (E.g. on the local network or a cloud service hosted by the device supplier)
4) Core systems authenticate device as factory configured device with basic authorization.
5) Human (operator, electrician, engineer or similar) performing the installation associates the physical/logical location (location as registered in the core systems referring to the point where the device is installed) with the MAC address, S/N or other agreed upon identifier of the device. This can be achieved through a mobile interface (e.g. laptop, smart-phone or tablet) where the installer selects the correct location and either reads the identifier from a bar-code, RFID or similar tag on the device or transfers the identifier from the mobile interface to the installed device using NFC or similar communication.
6) Device registers with the core systems providing its identifier for identification.
7) Once the identifier is available at both the installed device and at the core systems a service connection can be orchestrated between the configurable device and the appropriate configuration store.
8) Once the connection is made between the configuration repository and the identified device, the complete configuration containing the assigned task is transferred to the device and installed/executed.
9) Using the received configuration the device is automatically able to start performing its assigned tasks.
10) A message of success/failure is sent as an event to subscribed user interfaces.

C. Bootstrapping of Resource Constrained Devices

As an example of how the general procedure may need to be specialized to fit certain requirements, a more limited procedure has been designed and tested. The purpose of this is to show how a general procedure may be of use even though the application field is very diverse and it may be difficult to apply the original procedure explicitly in every case.

A zero configuration approach for IoT devices requires the use of Bootstrapping techniques. From the point of view of a wireless sensor and actuator (WSAN) node, the first time that a new node connects to the wireless network it only knows its own IP address and the IP address of the gateway; it has normally no information about other services in the network.

Bootstrapping is a pseudo-configuration service which provides a basic configuration of the mandatory and essential services that a node requires and needs during the boot process, examples are; configuration services, authentication services or device manager service.

As the only information that a node has after connect to the wireless network is the IP of the gateway, the Bootstrapping service must run on it and should use a predefined port (this is the unique predefined information). The bootstrapping request can include information about the IoT device to create a customized and optimized response for the device; this information can contain a serial number, a predefined ID, MAC address, internal software name, version, or other information able to identify the device or at less the device type.

After this process, the device should store all the information in a non-volatile memory, and use it in case the connection to the bootstrapping service goes down.

The penalty for the utilization of this technology regarding communication is a single request per boot but also it requires the implementation of a parser on the device, which can consume valuable memory on resource-constrained devices.

The following example is a bootstrapping profile encoded with JSON (Code 1), but it can be encoded with CBOR to reduce the packet size.

Code 1. Example of Bootstrapping for an IoT node encoded in JSON

```
{
  "auth": {
    "ip": "fdfd:0:0:0:0:0:0:0A",
    "port": 5683,
    "v": 1,
    "res": "/Authentication",
    "resAlt": "/Authorization"
  },
  "conf": {
    "ip": "fdfd:0:0:0:0:0:0:0B",
    "port": 5682,
    "v": 1,
    "res": "/Conf"
  },
  "dev": {
    "ip": "fdfd:0:0:0:0:0:0:0C",
    "port": 5681,
    "v": 1,
    "res": "/rd"
  }
}
```

D. Intended areas of configuration

To illustrated how the configuration approach may provide different possibilities for different areas of application, this section provides a few examples from some of the domains where Arrowhead is intended to be used.

1) Control code to PLC's and IoT controllers: Writing control code is one of the critical aspects of automation systems' engineering process. The traditional approach to program control systems does not fit well within the paradigm of Cyber-Physical Systems (CPS), where physical systems need to evolve in intimate and aligned correspondence with their virtual representation [27]. To address this, the Arrowhead approach propose a data-driven method for control code deployment, deployable on a number of devices and platforms with embedded data storage and processing capabilities, for the configuration of automated manufacturing systems.

The proposed architecture is composed of three types of elements (see Figure 3): i) data model that describes the system structure and the control behaviour, ii) logic engine that orchestrate system by interpreting description of the system defined within the data model, and iii) resource specific standard library Function Blocks (FBs) acting as an interface between the hardware (i.e. sensors and actuators) and the data model.

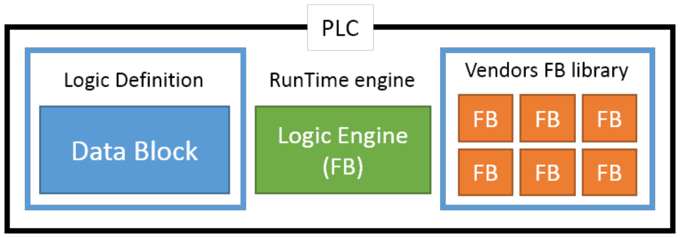

Fig. 3. Control software architecture

In this software architecture, the logic engine is configuration independent as for any system configuration it remains unchanged while the data model is configuration dependent as any change in the system, such as sequence, require reconfiguration of the data model. Resource-specific FBs can remain the same for different control or hardware configuration, but may need to be changed in some cases (e.g. vendor specific configuration, actuator type/configuration etc.).

The motivation for this software architecture is to dissociate the key elements required to achieve the overall device configuration and therefore dissociate the engineering processes that support various aspect of the device configuration (e.g. device firmware/logic engine update, system specific configuration change/update, etc.). This approach also allows generating the control code using standard library components (i.e. FBs), which are driven by the control logic defined in the manufacturing process simulation tools to enable seamless transition from virtual to physical system and feedback of runtime data to virtual system to facilitate data model calibration and analytics. As the control behaviour is defined in the data model, which can be accessed (i.e. read/write) in runtime, service visualization and process parametrisation can be attained using human-machine interface devices.

2) Configuration of sensors and actuators: Sensor and actuator nodes usually have two types of physical resources: sensors and actuators; and in the case of IoT systems based on Service Oriented Architecture (SOA) the configuration can also set the service's behaviour. Therefore, there are three different types of configurations.

1) Sensor
   - Sample rate
   - Inactive periods
   - Sensitivity
2) Actuator
   - Sensitivity
   - Active/Sleep
3) Service
   - Service composition
   - Filtering
   - Data compression
   - Output format
   - Triggering

3) Configuration of Building Automation Controllers: Building Automation Controllers (BAC's) use multiple physical resources: sensors, actuators and remote i/o units as well as being freely programmable with applications for monitor & control, communication, HMI and event handling. Therefore, there are many different types of configuration but they can be divided into functional configurations (programming, setup, services) and operational configurations (settings), the later often provided by services in a SOA environment.

IV. Use cases and Evaluation

The proposed approach is intended to be flexible and to provide different possibilities for scenarios likely to be encountered in different areas of automation. Some use cases have been collected to illustrate the benefits, and possible drawbacks of the approach.

A. *Use case: Building automation*

In the field of building automation it is common that many systems in one area are to use identical configurations, or that there are a few typical configurations that are used for a large number of systems. This may be the case for an area with apartment buildings that are all built during the same era and are owned by the same company, here it is beneficial for the owners from a maintenance and management point of view if the systems are as similar as possible.

In this case, and similar scenarios, a centralised Configuration store allows management and updates of all systems based on one configuration that can be rigorously tested and monitored for its first deployment. Once the initial deployment has proven successful it can be applied to all others with low risk of failure.

Additionally, a Configuration store that can keep track of configuration status will allow easier comparison of similar buildings using different configurations, in order to optimise or find flaws in the tested versions.

B. *Use case: Manufacturing industry*

Due to the growing need to manufacture highly innovative and customized products, efficient and quick adaptation of manufacturing systems to new product and production volumes is of significant importance. It has been established that one of the major obstacles in realising an efficient and reconfigurable production systems is the existing PLC control code development and deployment approach. The management of PLC devices configuration is currently completely dissociated from other engineering phases (such as process planning and mechanical engineering) and from the digital data set resulting from them.

WMG and FDS is developing a virtual engineering toolset, vueOne, and associated CPS oriented engineering methods that aim at filling this gap by providing data-driven control code generation capabilities (described in section III, D) directly from the vueOne virtual process planning module [27]. In this use case, an engineering scenario focusing on PLC devices configuration is investigated using Web service and Arrowhead Framework to enable direct deployment of control software to PLC devices to provide basis for dynamic and more distinctive configuration scenarios.

Unlike the classical method of PLC device configuration, which requires a direct connection between the PLC and a laptop running the vendor-specific programming tool, the PLC devices (or a server module linked to it) subscribes to the Configuration Store. Any changes in the control configurations are then directly passed from the Configuration Store to the subscribed PLC device when available. To capture the changes made to in the control code on the shop-floor, if any local changes are made in the control software, such as changing some parameters of the data model, the PLC device uploads the latest configuration to the Configuration Store to update the configuration database.

In an ideal configuration mechanism, the PLC device will retrieve updated configuration if/when available automatically. In practice however, the approach requires access to proprietary APIs from PLC vendors to allow download access to the PLCs. The Configuration Store consumer software is currently deployed on laptops that control engineers connect to PLCs in order to update or install control code. However, in case of embedded controllers the dynamic configuration can be achieved seamlessly using the Arrowhead framework due to their non-proprietary configuration mechanisms.

C. Use case: IoT devices

As the number of devices increase, so does the need for technologies for large scale management of systems including hardware devices and life cycle management. The issue of life cycle management, in particular configuration, is even more complicated when it comes to managing a very large number of resource constrained, wireless and battery powered devices in harsh environments.

In this use case, the authors have investigated how advanced configuration can be achieved in a very efficient manner in terms of communication overhead and energy usage.

The introduction of the IP technology for Wireless Sensor Networks (WSNs), now called Internet of Things (IoT) is today a hot topic, the power consumption of application's level protocols was a barrier which hindered a massive expansion of IoT; But in 2014, the standardization of CoAP [28] helped to develop IoT systems. The application of CoAP was a step forward and changed the behaviour of the WSAN nodes, from simplistic pull-based to more complex event-based communication. Nowadays, each node can provide services (resources) to other nodes, or to any server/client at Internet. This enables deployment of smart and efficient IoT systems with advanced features such as; service composition, event detection, low-power operation, high dependability, etc.

All these new features requires either run-time zero configuration and/or static configuration; In order to provide run-time configuration which is a much better approach for deploying larger networks, the framework must implement both bootstrapping and configuration services. The validation of these services for bootstrapping and configuration are a direct comparison between the benefits they provide versus their respective performance impact in terms of power consumption, communication overhead and memory usage.

The proposed Bootstrapping and Configuration services enable the following features:

- Low-power. The configuration of the previous WSNs was static; that means that the performance of each device was the same during the life-cycle. But now with Configuration service, the performance can be adaptive. All variables like sample rate, type of processing, triggers, etc. can be modified.
- Dependability and Robustness. The bootstrapping nature is dynamic, on each request, the bootstrapping service creates a customized response; Then if a service goes down, another one can replace it, just changing the bootstrapping parameters.
- Zero Configuration. With bootstrapping there are no fixed end-points on the device, one node could be deployed at any WSNs, and it will be able to establish a correct configuration.

The consumption of these two services (Bootstrapping and Configuration) is not excessive compared to others as Authentication, Authorization and Device Manager. The values in Figure 4 can change depending on the complexity of the device configuration; these values are based on experimental results of energy consumption. The benchmark configuration relies on measures of battery current and voltage externally to the device; these measurements are done using a 16-bit ADC at 1840 Hz to capture rapid events such as radio, wake-ups, etc. All these measurements are combined to 8 digital inputs that can be used to recognize in detail the power consumption of each software module. The selected IoT platform to do the test was a Mulle from Eistec AB, which is equipped with an ARM Cortex-M4 at 100 MHz micro-controller and an IEEE 802.15.4 transceiver. It has an on-board 2 MB of flash memory and 256 KB of internal memory on the micro-controller. The Mulle runs the open-source Contiki OS; so all taken measures are affected by running an OS on the same device without any isolation to get real condition data.

Regarding memory, the biggest cost is to parse the incoming configuration profile. For this experiment, the profile was encoded in JSON format, and the jsmn [29] C-API was used as the parser. The configuration memory usage represents the memory of configuring a single service. The bootstrapping memory usage also includes the JSON parser's ROM memory usage (see Table I).

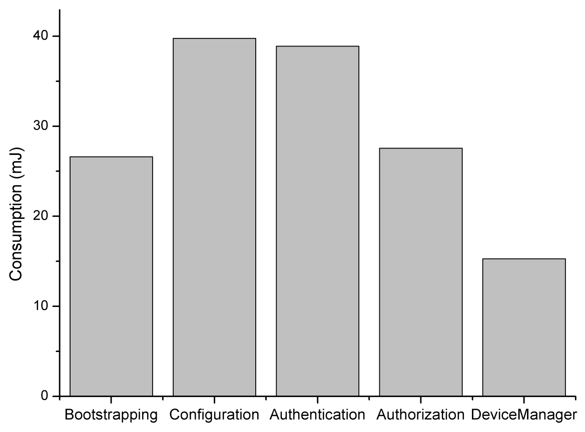

Fig. 4. Comparison of energy consumption between Bootstrapping, Configuration and other common IoT-WSNs services.

	Bootstrapping	Configuration
ROM (bytes)	1126+4382(parser)	840
RAM (bytes)	412	776

TABLE I

MEMORY FOOTPRINT OF BOOTSTRAPPING AND CONFIGURATION

D. Use case: Process industries

The process industry typically has comparatively static production lines where reconfigurability of the production, as commonly described for manufacturing industries, is not usually sought after. However, another scenario that may require significant configuration is that of replacement of devices, machines or groups of systems.

In the current environment of process industry automation systems, replacement of devices or larger systems generally takes place because of one of two reasons.

The first reason is that the existing one is broken, faulty or worn out and should be replaced with a new, but identical, item. This case is generally considered to be part of standard maintenance procedures, either reactive or proactive.

The second reason is that the existing item is too old and should be replaced with a newer part, this may be due to e.g. discontinued support from suppliers and difficulty to procure replacements, difficulty to attract and retain personnel with the required expertise in older systems or due to lack of technical features or functionality. This case is generally considered part of the long term investments in production systems and as such, replacements are generally planned carefully and in great detail.

While the conditions before the two cases are very different, especially with regards to the possibility to make preparations, as one is planned and the other is not, there are also some commonalities in that in both cases the functionality is expected to be exactly the same as before and that the time for the replacement and commissioning is usually very limited.

Traditionally these kind of replacement procedures have relied heavily on existing documentation and backup copies of control code and configurations, however these are not always kept up to date and may not be fully compatible with updated replacement devices. This leads to additional engineering work and may in unfortunate cases lead to unforeseen complications during the critical commissioning work.

As the Configuration store can be used to store the most current configuration for each device and the deployment procedure allows for a smooth introduction of new devices, an easier, quicker and cheaper replacement process is expected.

However, the Configuration store does not solve the issue with incompatible configurations or software version, but having a backup that is certainly of the most recent configuration will make engineering work easier and the re-commissioning less uncertain. If the device manufacturers can be persuaded to use open and standardized configuration files this would allow more possibilities for conversion and testing of updates ahead of the physical replacement.

V. DISCUSSION

The use cases presented are diverse and have significantly different requirements and potential benefits. The case presented from Building automation highlights the benefits from a management and maintenance point of view. The use case from a Manufacturing industry illustrates some of the strong potential once an approach like this is fully integrated in a complete engineering tool chain, something that can be expected to happen eventually in all areas with significant engineering requirements. However, it also illustrates some of the limitations that may occur when existing hardware is not yet capable to make use of new possibilities.

The IoT case illustrates the drawbacks that can be expected from using the approach for resource constrained devices. The cost incurred in terms of energy and memory consumption is not negligible, but for scenarios where a large number of devices are to be deployed and configured it is likely to be acceptable. At least when compared to the additional engineering time required to configure all devices individually prior to deployment. The use case from a Process industry shows that a structured, centralised management of device configurations can provide benefits to diverse elements of society. In addition, keeping the records of device configurations automatically updated is likely to remove some of the additional cost incurred when a device needs to be replaced or upgraded.

VI. CONCLUSION

In this paper, we have presented the Arrowhead Framework's approach for advanced configuration of systems and devices. The proposed approach is designed to be highly versatile while still being efficient for usage by resource-constrained devices. The approach has been implemented and tested on a resource-constrained wireless sensor and actuator platform.

Test results indicates that the overhead from performing bootstrapping and automatic configuration of a newly deployed device is negligible in terms of energy consumption and memory usage in relation to the energy spent by the device for sensing purposes during its lifetime. This shows that advanced run-time configuration is feasible even on small, battery-powered devices.

VII. Future work

Among the possible paths of future advances can be found several interesting options. Along the path of further implementation lies the tasks of prototype implementations for configuration of larger, less limited devices.

Further development could include implementations more integrated with engineering tools. This could be done either by implementing methods for managing the ConfigurationStore from existing, discipline related engineering tools, or by designing an engineering tool centred around the ConfigurationStore and associated processes. The former implementation would allow for a smooth engineering and deployment process within the specific domain, while the latter has an advantage of spanning all affected domains easing e.g. management and maintenance of multi-domain facilities.

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] (2016). [Online]. Available: https://www.isa.org

[2] B. Scholten, *The Road to Integration: A Guide to Applying the ISA-95 Standard in Manufacturing*. ISA, 2007.

[3] (2016). [Online]. Available: http://isa99.isa.org/ISA99%20Wiki/Home.aspx

[4] K. Staggs and et.al., "ISA $62443-4-2$ security for industrial automation and control systems technical security requirements for iacs components," ISA, Draft Standard Draft 2 edit 4, July 2015.

[5] 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.

[6] (2016). [Online]. Available: http://www.socrades.net

[7] 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.

[8] Wikipedia, "FiWare — wikipedia, the free encyclopedia," 2016, [Online; accessed 21-April-2016]. [Online]. Available: https://en.wikipedia.org/w/index.php?title=FIWARE&oldid=711836994

[9] H. Derhamy, J. Eliasson, J. Delsing, and P. Priller, "A survey of commercial frameworks for the internet of things," in *Emerging Technologies & Factory Automation (ETFA), 2015 IEEE 20th Conference on*. IEEE, 2015, pp. 1–8.

[10] T. Erl, *SOA Principles of Service Design (The Prentice Hall Service-Oriented Computing Series from Thomas Erl)*. Upper Saddle River, NJ, USA: Prentice Hall PTR, 2007.

[11] 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.

[12] 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.

[13] 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.

[14] 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.

[15] R. Kyusakov, P. P. Pereira, J. Eliasson, and J. Delsing, "Exip: a framework for embedded web development," *ACM Transactions on the Web (TWEB)*, vol. 8, no. 4, p. 23, 2014.

[16] 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.

[17] J. Delsing, F. Rosenqvist, O. Carlsson, A. W. Colombo, and T. Bangemann, "Migration of industrial process control systems into service oriented architecture," in *IECON 2012*. IEEE, 2012.

[18] 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.

[19] 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.

[20] S. Karnouskos, A. W. Colombo, F. Jammes, J. Delsing, and T. Bangemann, "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.

[21] J. D. ed. (2016) Arrowhead framework wiki. [Online]. Available: http://forge.soa4d.org/arrowhead-f/wiki

[22] S. Hodek and J. Schlick, "Ad hoc field device integration using device profiles, concepts for automated configuration and web service technologies: Plug amp;play field device integration concepts for industrial production processes," in *Systems, Signals and Devices (SSD), 2012 9th International Multi-Conference on*, March 2012, pp. 1–6.

[23] D. Cachapa, R. Harrison, and A. Colombo, "Configuration of SoA-based devices in virtual production cells," *International Journal of Production Research*, vol. Volume 49, no. Number 24, pp. 7397–7423, 2011. [Online]. Available: http://wrap.warwick.ac.uk/54497/

[24] L. Dürkop, H. Trsek, J. Otto, and J. Jasperneite, "A field level architecture for reconfigurable real-time automation systems," in *Factory Communication Systems (WFCS), 2014 10th IEEE Workshop on*, May 2014, pp. 1–10.

[25] C. Perera, A. Zaslavsky, M. Compton, P. Christen, and D. Georgakopoulos, "Semantic-driven configuration of internet of things middleware," in *Semantics, Knowledge and Grids (SKG), 2013 Ninth International Conference on*, Oct 2013, pp. 66–73.

[26] F. Blomstedt, L. L. Ferreira, M. Klisics, C. Chrysoulas, I. M. de Soria, B. Morin, A. Zabasta, J. Eliasson, M. Johansson, and P. Varga, "The arrowhead approach for soa application development and documentation," in *IECON 2014 - 40th Annual Conference of the IEEE Industrial Electronics Society*, Oct 2014, pp. 2631–2637.

[27] R. Harrison, D. Vera, and B. Ahmad, "Engineering Methods and Tools for Cyber-Physical Automation Systems," *Proceedings of the IEEE*, vol. 104, no. 5, pp. 973–985, May 2016.

[28] Z. Shelby, K. Hartke, and C. Bormann, "The Constrained Application Protocol (CoAP)," RFC 7252 (Proposed Standard), Internet Engineering Task Force, Jun. 2014. [Online]. Available: http://www.ietf.org/rfc/rfc7252.txt

[29] "Jsmn c json parser," https://github.com/zserge/jsmn, updated: 2016-01-15.