Chapter 14
Service-Oriented Architectures for Interoperability in Industrial Enterprises

Ahmed Ismail and Wolfgang Kastner

Abstract This chapter focuses on the technological aspects involved in developing a service-oriented solution for interoperability in the context of cyber-physical production systems (CPPS). It addresses the typical state of industrial enterprises and the core technologies currently available for the development of a service-oriented (SO) solution for agile environments. The chapter therefore discusses features of the service-oriented paradigm as well as aspects related to enterprise and network architectures, constraints, and technologies to discern the current challenges facing modern enterprises. The chapter also explores the service-oriented reference architectures of recent EU projects to highlight their main characteristics. Finally, their respective realizations are decomposed to discern the connectivity strategies and standards employed by each to achieve an interoperability-focused technology stack for the operation of agile and flexible industrial plants.

Keywords Horizontal integration • Interoperability • Service-oriented architectures • Technology stack • Vertical integration

14.1 Introduction

The increasing rapidity of change in the environmental factors of modern enterprises requires that they be able to adapt to external and internal stimuli over increasingly shorter timescales (Corrêa, 2001). Industrial enterprises must be agile to withstand and thrive in such dynamic environments. The defining characteristics of their systems should include having "easy access to integrated data whether it is customer driven, supplier driven, or product and process driven", "modular production facilities that can be organized into ever-changing manufacturing nodes", and "data that is rapidly changed into information [for the expansion of] knowledge", amongst other things (Choudri, 2001). Together, such features may fuel a successful agile enterprise.

A. Ismail (✉) • W. Kastner
Institute of Computer Aided Automation, Technische Universität Wien, Wien, Austria
e-mail: aismail@auto.tuwien.ac.at; k@auto.tuwien.ac.at

S. Biffl et al. (eds.), *Multi-Disciplinary Engineering for Cyber-Physical Production Systems*, DOI 10.1007/978-3-319-56345-9_14

From a technical perspective, the pursuit of such characteristics can be supplemented using a number of techniques from the domain of service-oriented architectures (SOA). This is a field that is concerned with the creation of modular IT and productions systems that enhance an enterprise's capabilities for information exchange, technological independence, and component reuse. The result would effectively be an industrial environment of operational flexibility and responsiveness (Valipour et al., 2009).

This chapter focuses on service-oriented architectures and how they may be applied in current enterprises to achieve flexibility, agility and interoperability, all within the context of research question ***RQ I1*** of Chap. 1. As such, it will provide an understanding of the technical state of present industrial enterprises and detail the characteristics of the SO approach in order to highlight the competitive advantage possible through service orientation. A sizeable part of this chapter is dedicated to presenting preliminary SO reference architectures delivered by major European Union research projects. Respective realizations of these architectures will be discussed to underline their choices in technologies and their delivered technical innovations.

14.2 Technical Features of the Industrial Enterprise

Typically, an industrial enterprise undergoes functional segmentation creating layers that distinguish between the components of the process, control, operations, business, and Internet-based systems present in or interacting with the enterprise. Normally, a demilitarized zone (DMZ) is also included to manage access from the uppermost business and enterprise layers to the lower process-focused layers' network and data. This kind of segmentation is done to increase the manageability and security of the enterprise. However, further impositions on the enterprise exist due to the physical and logical constraints of the enterprise's assets. Physically, devices may be connected using legacy serial interfaces (e.g. EIA-232/422/485), fieldbus systems, and wired and wireless Ethernet and IP-based technologies. Although devices with both interface types may be used to physically bridge the two networks together, the protocols may have different demands in real-time (RT), bandwidth, latency, and other communication-related requirements. At the messaging layer, these protocols may also differ in their message formats and exchange mechanisms, as well as in other features. The process of bridging together these various systems requires special devices and techniques, designed and implemented carefully to safely allow them to share data (Ismail and Kastner, 2016; Knapp, 2014).

By convention, there are two approaches for managing this heterogeneity and technical complexity, namely tunneling and translation. Tunneling involves the use of routers for the encapsulation of one protocol's data inside the payload of another and treating the channel as a transparent communication medium. Translation, on

the other hand, uses gateways as intermediaries to carry out data mappings on behalf of communicating devices to allow them to exchange information using their native protocols. Each of these methods has its own drawbacks in relation to capabilities and implementation complexity, and is often considered to be costly in engineering efforts (Sauter et al., 2011).

These costs may be compounded by the technological choices and implementations of enterprise infrastructures, which commonly use monolithic applications to achieve their functional goals. The resulting system involves many implicit and explicit dependencies (i.e., to a technology stack) that introduce stiff resistance to change. In such a case, the enterprise cannot be described as agile or flexible. In fact, it is properties such as these and their implications that have become major arguments used by the proponents of service oriented architectures to effect infrastructural changes in enterprises (Krafzig et al., 2005).

Take for example the concept of a service-oriented CPPS. CPPS are physical and computational resources that are tightly bound in coordinated and controlled relationships and embedded in a socio-technical context. The functionalities originally addressed by monolithic applications may be decomposed and distributed across the system's member devices. That is, by applying the service-oriented design pattern, a large business problem can be fragmented into smaller problems that may then be solved using a number of small and related units of logic, termed services, rather than through a single monolithic application. Distributing these services across the system would create networks of smart devices that are inherently resilient due to their lack of dependence on any central component. Furthermore, as services are typically designed with standardized interfaces, system-wide interoperability is guaranteed. The internals of these services, such as how they are implemented, or what technology they use is hidden behind the service interface therefore also affording the system technological independence and flexibility. Services are also typically designed with functional agnosticism to allow for their reuse, reducing future application development efforts. Together, the concepts that define the SO approach, all of which are summarized from Valipour et al. (2009), Erl (2009), and Erl et al. (2014) and presented in Table 14.1, afford an enterprise the agility it pursues and minimizes the need for integration (Krämer, 2014; Rubio, 2011; Erl, 2009; Valipour et al., 2009).

For reasons such as those mentioned above, several research projects have pursued and outlined reference architectures for highly interoperable industrial environments based on SOAs. These efforts have been extensively documented to facilitate future system implementations. In the coming section, we highlight a number of these projects, summarizing and evaluating their reference architectures to give a concise understanding of their details.

Table 14.1 Features of SOAs (Valipour et al., 2009; Erl, 2009; Erl et al., 2014)

Characteristic	Sub-characteristic	Explanation
Discoverability		Services are supported using metadata that allows them to be discovered and interpreted
Modularity	Modular decomposability	The equivalent concept of functional decomposition as applied to modules
	Modular composability	The ability to create a software service or system by freely combining reusable services
	Modular understandability	The function of a service should be comprehensible without requiring knowledge on any other services
	Modular continuity	A service interface should conceal service implementation details to allow changes to the service to occur without them requiring changes in other services
	Modular protection	Perimeterisation of modules to prevent the cascading of faults unto other services
Interoperability		Ensured ability for different modules to communicate with each other
Loose coupling		Appropriately defined service contracts that increase the independence of services from their implementations and from each other
Location transparency		The decoupling of a service from a specific location allowing dynamic service lookups and runtime binding that enhance the system's flexibility, availability and performance
Composability	Application	The piecing together of services and their orchestration using application logic to achieve specifically set goals
	Service federation	The aggregation of services under a single service representation
	Service contracts	The explicit definition of a service's features and parameters as contractual terms and conditions in a granular form accessible by service requesters. This may include the definition of supported data types, data models, policies and other features that declare a service's interaction requirements
	Service orchestration	Service execution as part of an application should be sub-transactional and not permitted to perform data commits. This allows the system to rollback to the pre-transactional state in case of service failure

14.3 Service-Oriented Architectures and the Industrial Enterprise

In this section, we look at a total of five service-oriented reference architectures that resulted from collaborations between research, vendor, and user organizations These are the Internet of Things at Work (IoT@Work), Embedded systems Service-based Control for Open manufacturing and Process automation (eScop), Production Logistics and Sustainability Cockpit (PLANTCockpit), ArchitecturE for Service-Oriented Process—Monitoring and Control (IMC-AESOP), and Arrowhead Framework projects; the inferred or explicitly stated purpose of each of these projects is summarized in Table 14.2. This list of projects is not comprehensive as there exists a larger number of SOA-driven projects both old, such as SOCRADES and SODA, and new, such as ProSEco and CREMA, that are not covered in this chapter. Nor is it the point of this chapter to give such a viewpoint, but the purpose here is to bring attention to the main features of a sample of SO reference architectures to underline their strategies for the modernization of industrial enterprises. As such, our analysis is limited to projects that started and completed within the period of 2010–2016. Arrowhead does not meet this timeline, as it is scheduled for completion in 2017, however, an exception is made in this case as, with a 77-member consortium and 69

Table 14.2 A summary of project durations and objectives

Name	Duration	Objective
IoT@Work	Jun 2010–Jun 2013	Use IoT technologies to decouple application/control programming from the network, enable communication-centric plug & work capabilities, and enhance the system security (Rotondi et al., 2013)
PLANTCockpit	Sept 2010–Dec 2013	Creating a SO and centralized plant-wide human-machine interface (HMI) (PLANTCockpit Consortium, 2011a)
IMC-AESOP	Sept 2010–Dec 2013	Using SO approach for supervisory control and data acquisition/distributed control system (SCADA/DCS) in large-scale process control systems (Colombo et al., 2014a)
eScop	Mar 2013–Feb 2016	System integration using ontology based knowledge-management, embedded devices, and SOA (eScop Consortium, 2013)
Arrowhead framework	Mar 2013–Feb 2017	Providing a SO technical framework for cooperative automation in technologically heterogeneous systems (Blomstedt et al., 2014b)

million Euros in funding, it is one of the largest EU projects in the industrial domain (Nagorny et al., 2014).

14.3.1 IoT@Work

The IoT@Work project represents its reference architecture using layers and planes. In terms of the former, three layers are used; the physical, abstraction, and composite service layers. The first of these, the physical layer, is the physical world and is therefore composed of physical devices. The second layer is an abstraction of the physical devices as resources and services. In the context of the IoT@Work architecture, a resource is an object representing a specific physical or virtual element, while a service gives access to a resource by specifying the type, identifier and interface. Effectively, a single device may be represented using one or more resources and services. The third and final layer is that of composite services. These group together the elements of the second layer to hide their complexity and deal with context, contention over resources, and access rights. It is this third layer atop which applications such as event notification, complex event processing (CEP), network access control (NAC), and controller I/O applications run (Rotondi et al., 2013).

To address the functional aspects of these three layers, the IoT@Work project defines a set of core services, listed and defined in Table 14.3, and organizes the large number of functional components they are composed of into three planes; the communication, security, and management planes. The communication plane is concerned with the orchestration of network resources and communication to resolve access contention issues and provide support for Quality of Service (QoS) guarantees. The security plane, as the name implies, manages and integrates security into the overall system. Lastly, the management plane, attends to device, service, and configuration management with a focus on their inter-relations (Rotondi et al., 2013).

14.3.2 PLANTCockpit

The PLANTCockpit system architecture is composed of an internal and external system. The external system refers to the data sources connected to the PLANT-Cockpit using proprietary or open interfaces, such as an Enterprise Resource Planning (ERP) system, OPC server, or sensors & actuators. As for the internal system, this is made up of five layers; the system connector, function engine, persistence, visualization engine, and presentation engine layers (PLANTCockpit Consortium, 2011a).

The first of these, the system connector layer, is primarily concerned with interfacing with external data sources. It provides the configurable *adapter* modules

Table 14.3 IoT@Work core services (Rotondi et al., 2013)

Core service	Explanation
Event notification service (ENS)	A common functional component that collects and distributes events
ENS access request broker (ARB)	A broker between ENS clients attempting to access namespaces and the ENS AS
ENS authorisation service (AS)	Decision point for access requests sent to the ENS ARB
Policy decision point (PDP)	Evaluates the status of capability tokens and policies to approve or refuse access requests
Revocation service	Manages capability revocation requests and capability revocation life cycles
ENS namespace management service	A service for the management of hierarchical structures used for the organization of event publishing
Slice management system	A three part service consisting of a communication service interface (CSI), slice enforcement point (SEP) and slice manager. Used for the creation of a 'slice'[a]
Embedded application configuration service	Provides devices with the configurations required by their applications
Directory service (DS)	Stores device information in an ontology-based DS data model
Orchestrated management	Orchestrated management authoring support: a lightweight algorithm and API
	Orchestrated management scheduling service: algorithms to produce management plans and schedule operations
	Management services: a wrapper around existing operations in the three planes so that they may be used and executed in Orchestrated Management scenarios
	Context services: capture constraint values to provide context. May be a parameter of the manufacturing execution system (MES) or ERP system
Complex event monitoring service	Responsible for the verification of rule compliance to allow the system to meet safety and security goals

[a]A slice is a virtual network with QoS guarantees and policies

required to allow the PLANTCockpit to access and communicate with these sources. Due to their configurability, an *adapter manager* is included in the architecture to oversee the entire life cycle of adapters. As for the external data structures acquired through the adapters, these are transformed to an internal data structure using a *mapper* module. Finally, the layer uses two generic components, the *subscriber* and *publisher*, to query the external systems via the adapters and push the data retrieved by way of the adapter and mapper components to the function engine layer, respectively (PLANTCockpit Consortium, 2011a).

The function engine layer, receiving these data, provides a platform atop where analytics and functions may be executed. It is based on the concept of *function blocks*, which, inspired by the object oriented paradigm and the IEC 61499 standard, are reconfigurable and encapsulated blocks of program code with clearly defined interfaces to allow for reuse and composability. These blocks' life cycles are managed using a *function manager*, while a *pub/sub broker* (publisher/subscriber) provides them with secure, reliable, and event-driven mechanisms through which they may communicate with each other (PLANTCockpit Consortium, 2011a).

Any data relevant to the function engine or any other layer's workings are managed and stored using the persistence layer. This subsystem is composed of three components; the *data persistence manager*, *configuration repository*, and *data repository*. The manager administers the storage, archiving, retrieval, and deletion of data. The configuration repository maintains all of the data needed to configure the internal components of the PLANTCockpit system at design and run time. Finally, the data repository stores all of the data required by analysis processes in the PLANTCockpit system. It includes a cache that can temporarily store data to improve the system performance, and a more permanent store that archives historical data (PLANTCockpit Consortium, 2011a).

Finally, any data to be presented via the HMI interface is prepared for visualization using the visualization engine layer. It consists of a service engine which is an aggregation of a runtime and design engine. The former contains the configurable user interface (UI), while the latter configures the interface using a composition of *building blocks* (visualization elements) and their associated data points. A *building block browser* and *function block browser* is used to make all possible building blocks and all available data points provided by the data persistence manager accessible by the design engine for the UI's configuration, respectively. Finally, a *data provider* component subscribes to the Pub/Sub Broker for data and events that it delivers to the runtime engines. The presentation engine layer, which is composed solely of a *presentation runtime engine*, then graphically presents the configured building blocks (PLANTCockpit Consortium, 2011a).

14.3.3 IMC-AESOP

As opposed to the PLANTCockpit framework, the IMC-AESOP architecture attempted to provide a generic architecture to support multiple applications, with

the HMI only being one of these applications. As such, the framework is in fact a behemoth of services, service groups, and interactions presented using both natural language and semi-formal descriptions based on the Fundamental Modeling Concepts (FMC) graphical notation. The abridged description of the framework's components are shown in Table 14.4 (Colombo et al., 2014b).

Based on the architectural overview given in Colombo et al. (2014b), the framework differentiates between four system components. The first of these is the user roles, which designates users *business*, *operations*, *engineering*, *maintenance*, or *training* roles. These roles interact with or impact the architecture directly or indirectly as they take part in plant processes. The second system component is that of the service groups themselves. These act as the glue binding together the user roles, the external systems (the third component), and the plant data itself (the fourth) (Colombo et al., 2014b).

14.3.4 eScop

The eScop project is composed of five layers, a *physical* (PHL), *representation* (RPL), *orchestration* (ORL), *visualization* (VIS), and *interface* (INT) layer. The PHL is concerned with the physical equipment in the eScop system and therefore provides device and service descriptions. The information provided by the physical layer is consumed by the RPL, which is responsible for knowledge representation. Syntactic and semantic service descriptions based on the service implementations are mapped and stored in the RPL. The ORL, which coordinates and executes service compositions, in addition to requiring service descriptions from the RPL, may also need input from the physical layer for it to be able to successfully orchestrate the execution of services. The VIS, configured by the RPL, then provides an interface for the user to interact with the system data accessible via the PHL. Finally, the INT acts as the entry point for external systems and services in the eScop architecture and is functionally concerned with the provision of technology adapters and access control measures (Iarovyi et al., 2015b).

As for the features of the reference architecture, the system's services define concrete technologies for implementation. To exemplify, the various components of each of the layers are as follows.

Starting from the bottom up, the PHL consists of an *I/O module*, *runtime core*, and *Web Services (WS) toolkit*. These support connections to the physical devices, the definition of applications on controllers, and provide the devices with web services and notification mechanisms, respectively (Iarovyi et al., 2015b).

The RPL achieves its goal of knowledge representation using an *ontology service*, a set of functions, and the ontology itself. The ontology service is in fact made up of four modules: *device registration*, *visualization provider*, *ontology manager*, and *service handler* modules. These handle device registration and deregistration in the ontology, assist with visualization, provide an interface for the

Table 14.4 IMC-AESOP service groups and services (Colombo et al., 2014b)

Service group	Component services	Explanation
Alarms	Alarm configuration Alarm & event processing	Defines, maps, filters, and aggregates alarms based on the principles of CEP. Supports simple alarms. May be time and/or event/alarm-triggered
Configuration & Deployment	System configuration service Configuration service Configuration repository	Responsible for the configuration, deployment, and enforcement of configurations on the various plant elements. The model repository service provides it with a structure for saving hierarchical configuration structures of nodes. The service may also manage the versioning of services and the instantiation of plant meta-models
Control	Control execution engine	Typically a distributed service, it can execute process models and configurations and support the online reconfiguration of processes and hot-standby redundancy
Data management	Event broker Data consistency Actuator output Historian Sensory data acquisition	Responsible for "data retrieval, consistency checking, storage, searching", basic eventing, and actuator output control. The service connects data producers with higher level services and provides methods for mapping data to the appropriate data models and ontologies
Data processing	Filtering Calculation engine CEP service	Provides services for basic filtering, CEP and calculations. The CEP engine allows for low-latency analysis of event data. It provides a management interface that allows for the creation, update or removal of rules used for processing events. The calculation engine supports users in executing numerical or logical operations over process data
Discovery	Discovery service Service registry	Supports the discovery of system components by type and location. Uses a registry to support the discovery of services implemented using technologies without inherent discovery capabilities, and to allow for discovery by remote entities where multicast and broadcast based discovery would be of limited usefulness
HMI	Graphics presentation	Provides a generic web interface where graphical tools may be embedded
Integration	Composition service Service mediator Gateway Business process management and execution service Model mapping service	Responsible for ensuring interoperability between heterogeneous components using translation and encapsulation. Also serves as a platform for the execution of business processes as service compositions and their presentation as higher level services

Lifecycle management	Code repository Lifecycle management service	Covers "aspects such as maintenance policies, versioning, service management and also concepts around staging (e.g. test, validation, simulation, production)". It also contains a code repository service which maintains the source code of services to allow for service maintenance, deployment, upgrade, and other functions
Migration	Infrastructure migration solver Migration execution service	Responsible for the migration of legacy systems to the SOA-based approach by identifying system dependencies, offering migration strategies, and executing them
Mobility support	Mobile service management	Concerned with the management of mobile assets and so is responsible for asset tracking, address mapping, data synchronization, and similar supporting functions
Model	Model management service Model repository service	Consists of generic services for the management of models, a repository to structure these models, and an interface to the repository
Process monitoring	Monitoring service	Provides HMIs with an entry point into the system. Responsible for gathering information from the physical process using other system components and semantically enriching it, and for handling alarms and events
Security	Security management service Policy management service	Enforces and executes security measures for confidentiality, authentication, and other features. Also defines and manages security rules/policies for identity or role-based access control and for the definition of identity federation
Simulation	Constraint evaluation Simulation scenario manager Simulation execution Process simulation service	Is connected to almost every other service group as it simulates systems and their processes, evaluating constraints and simulating execution. It consumes other systems' exposed simulation endpoints to imitate characteristic system performance and behavior features
System diagnostics	Asset diagnostics management Asset monitor	Concerned with monitoring the status and health of plant assets and mainly used for maintenance and planning
Topology	Naming service Network management service Location service	Allows for reporting and management on the system's physical and logical features. It includes domain name service (DNS) functionality, device discovery and management, and asset location services

configuration or editing of the model, or manage the RPL's connections to the various components of other layers. The governance of access to the ontology in the triplestore, on the other hand, is done by the RPL's *SPARQL query factory* and *ontology connector* internal functions, and it is suggested that, once secured, SPARQL-over-HTTP may be used for these factors. As for the ontology itself, this in fact is stipulated as being the eScop Manufacturing Systems Ontology (MSO), which is a proprietary component created by one of the designing members of the architecture (Iarovyi et al., 2015b; eScop Consortium, n.d.(a)).

The ORL coordinates the various components in the architecture using a *service composer* and an *orchestration engine*. The former maps process definitions to configurations applicable to the system, and the latter executes them (eScop Consortium, n.d.(a)).

The VIS aims to allow for flexible and generic graphical interfaces. For this it needs a *dynamic composition* module, a *symbol library*, and *visualization agent(s)*. Together, the VIS is able to map descriptions from the RPL to visualization elements from the symbol library which are then transformed by the agent(s) into web pages that can be displayed using a web browser (Iarovyi et al., 2015b; eScop Consortium, n.d.(a)).

14.3.5 Arrowhead Framework

Finally, the Arrowhead project divides its framework into three parts that are *design guidelines*, *documentation guidelines*, and a *software framework*. The first provides a description of design patterns for making legacy or newly created application systems compliant with the Arrowhead Framework. The documentation guidelines provide templates for the description of services, systems and system-of-systems. As for the software framework, this is the main concern of this section and is described in more detail below (Blomstedt et al., 2014b).

The software architecture defines a grouping of core services. These services are meant to support communication exchanges between domain-specific application services. These core services and systems are effectively divided into three groups: *Information Infrastructure* (II), *Systems Management* (SM), and *Information Assurance* (IA). II provides service descriptions and information on how to connect to services and systems. SM core services are concerned with orchestration and system-of-systems composition. Finally, those of IA address security and safety factors in information exchange processes. The categorization of core services and systems identified by the framework under these three groups is shown in Fig. 14.1 (Blomstedt et al., 2014a,b).

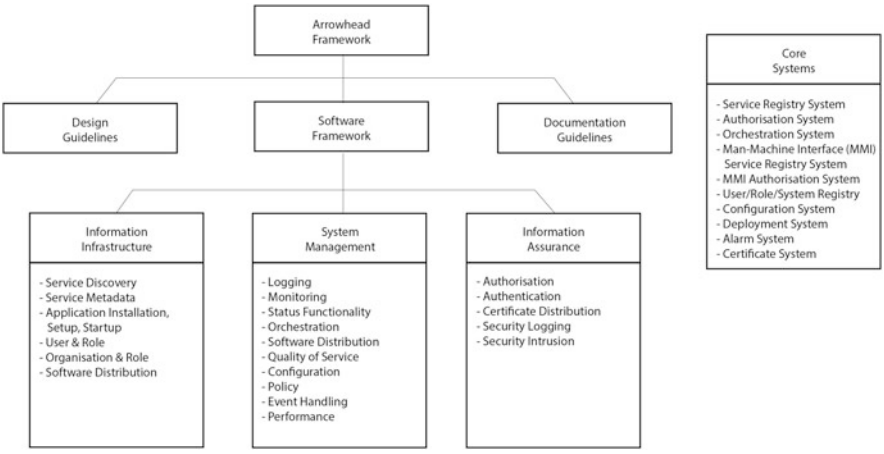

Fig. 14.1 The Arrowhead framework (Blomstedt et al., 2014b; Varga et al., 2014)

14.4 Realizations of the Reference Architectures

So far, the reference architectures have been described in an abstract manner. This portion of the chapter inspects the technology stacks implemented by each of the architectures. For comparability, we segregate these technologies into categories that address the various functional aspects addressed by all of the architectures. We provide brief descriptions on both the mature and novel technologies implemented for each category to give a succinct overview of the technical properties of each project in the pursuit of achieving interoperability.

14.4.1 Service Discovery

An essential aspect present in any service-oriented architecture is that of service discovery. Due to the close association of the device profile for web services (DPWS) with SOAs, the use of the web services dynamic discovery (WS-Discovery) specification is common. Four out of the five architectures, excepting eScop, either directly implemented or discussed methods to allow for the use of the WS-Discovery protocol.

The WS-Discovery protocol is based on the use of multicast messages (typically SOAP-over-UDP) to announce or probe for services using specially crafted eXtensible Markup Language (XML) documents. Announcements operate using multicast *Hello* and best-effort *Bye* messages. Likewise, *Probe* and *Resolve* messages are also multicast; the former is used to locate services based on service types and/or scopes, while the latter searches for a specific service by name. The use of a *Discovery Proxy* is encouraged to allow for the active suppression of multicast traffic in the network.

The specification also endorses the caching of multicast service advertisements to incur further savings. Finally, with respect to securing the discovery process, the specification does not require, but recommends the use of unique XML signatures and a number of other properties to mitigate against a variety of attacks (Bullen et al., 2009).

The IMC-AESOP is one of the architectures implementing the WS-Discovery protocol directly, for example, to allow Service Bus instances to discover each other (Nappey et al., 2013). However, one of the main contributions of IMC-AESOP hinges on its bridging of DPWS with the industry-focused OPC Unified Architecture (OPC UA) standard. The architecture therefore presents concepts for supplementing OPC UA's discovery mechanisms using WS-Discovery. To elaborate, the OPC UA discovery protocol requires the use of a discovery server. The address of the server must be known beforehand by participating OPC UA clients and servers. The IMC-AESOP approach presents two methods for auto-discovery in OPC UA systems using WS-Discovery. The first involves the use of WS-Discovery to allow OPC UA clients and servers to automatically find the OPC UA Discovery Server, while the second approach involves replacing the OPC UA Discovery Server with the WS-Discovery protocol to allow OPC UA clients to find OPC UA servers directly (Bony et al., 2011).

A second core technology in IMC-AESOP is the *Constrained Application Protocol* (CoAP). Identified as a suitable protocol for device-level integration of constrained devices, such as those belonging to wireless sensor networks (WSN), the IMC-AESOP approach discusses a reliance on the discovery mechanisms of CoAP, CoAP multicast, and the Constrained RESTful Environments (CoRE) Resource Directory (RD), for the location of services and resources hosted on resource-limited clients (Eliasson et al., 2013; Kyusakov et al., 2014).

As for PLANTCockpit, as the architecture is dependent on the concepts of adapters to interface with the various systems and function blocks, the discovery mechanism employed is dependent on the system being interfaced. For example, the system implements a DPWS adapter to allow for the discovery of DPWS devices. The adapters themselves, however, are implemented as function blocks based on the concepts of IEC 61499 function blocks. The identification of FBs depends on FB Service Interfaces, and these are implemented using the OSGi framework. As such, although not explicitly stated, it may be the case that the implementation depends on OSGi's service registry to register, get, or listen for services (Iarovyi et al., 2013; Dennert et al., 2013).

Although the IoT@Work approach explored and compared the WS-Discovery and OPC-UA's discovery mechanisms, it did so within the context of auto-configuration. Instead, the IoT@Work system uses a configurable RESTful Directory Service (DS) as a form of service registry. Devices interact with the DS using a RESTful API to retrieve, submit or delete device and service information using HTTP GET, PUT, POST and DELETE requests. The RESTful nature of the service allows every service to be modeled as a URL-accessible resource. The system also supports the use of QR codes and NFC tags for the identification of devices (Rotondi et al., 2013).

Similar to IoT@Work, the eScop project generates its own discovery mechanisms. Discovery here is based on the multicasting of Hello, Bye and Probe messages, and in this respect, it is similar to the WS-Discovery specification (Iarovyi et al., 2016). Inspection of the source code,[1] however, shows that the protocol does not, in fact, follow the WS-Discovery specification. This is as a number of critical differences exist, such as the use of JavaScript Object Notation (JSON) encoding for messages, and of multicast IPs and ports different than those stipulated for use by WS-Discovery.

Finally, the Arrowhead project defines three approaches for service discovery. The first is a service registry functionality based on the Domain Name System (DNS) and DNS Service Discovery (DNS-SD). Effectively, DNS is a hierarchical database mechanism that can store any kind of data and DNS-SD is a method for specifying how DNS resource records may be named, structured, and browsed. These records may be accessed using unicast DNS requests or multicast DNS (mDNS). The Arrowhead framework applies mDNS for constrained devices, such as those belonging to WSNs. The second and third approaches for discovery in the Arrowhead projects are based on the use of XML-over-HTTP and JSON-over-HTTP for RESTful web services. The former uses a DNS protocol specific to Arrowhead to allow for service discovery and the retrieval of service and data descriptions. The JSON-over-HTTP approach is marked to be done and an implementation still remains to be published. The framework does however discuss the prospect of implementing a translation service for integration between the XML/JSON and DNS-SD registry systems (Delsing, 2015; Cheshire and Krochmal, 2013; Arrowhead Consortium, 2014a; Blomstedt and Olofsson, 2016b; Blomstedt, 2016a,b; Blomstedt and Olofsson, 2016c).

14.4.2 Service Description

For this subsection, we focus specifically on the aspect of service contracts as defined in Table 14.1. The goal of service contracts is to define a minimal level of interoperability and thereby reduce the need for integration. It may do so by making available definitions of the service's functionality, data model, data transfer mechanisms and encodings, and policies for security and quality of service, amongst other things. Naturally, these contracts themselves need to be interpretable by all services available in the registry or operating environment. Similar to what was the case for service discovery, a web services technology, the Web Services Description Language (WSDL), is employed by a number of projects (Erl, 2008).

The WSDL language is a machine-readable XML-based language for the definition of interfaces. It is capable of describing all of a service's operations, the data required and output by each operation, and their respective data types.

[1] http://www.escop-project.eu/tools/.

It may also provide addressing and networking information to support inter-service connectivity. Both of the IMC-AESOP and PLANTCockpit projects use WSDL files for service descriptions. However, for describing sensor services, the IMC-AESOP project uses the JSON, XML and EXI-compatible Sensor Markup Language (SenML) (PLANTCockpit Consortium, 2012c; Colombo et al., 2014b).

The eScop project also employs a WS-based approach and develops a WS-enabled Remote Terminal Unit (RTU) titled the eScopRTU. Within the eScopRTU, all services are IEC 61131 Structured Text Language (STL) functions that are used to execute operations on resources. They are RESTful, hypermedia-driven, accessible via a REST API, and the API documentation, read "service descriptions", are created using Swagger. Swagger is a specification for the definition of language-agnostic, human and machine-readable representations of RESTful APIs. The specification requires that the API be described using either JSON or YAML[2]; the resulting files may then be processed by tools that can generate clients in a variety of languages. The Swagger ecosystem also includes tools to display and test the API. Swagger has since been renamed the OpenAPI Specification (OAS) (Faist and Štětina, 2015; Ratovsky et al., 2014).

IoT@Work, as previously mentioned, explored the prospects of auto-configuration using WS-Discovery and OPC UA. Part of the procedure outlined involves the acquisition of service descriptions. With WS-Discovery, this was achieved by having metadata on a DPWS-enabled device retrieved by its controller using the WS-Transfer protocol. The metadata that may be included is defined as part of the WS-MetadataExchange specification. This metadata would allow the service to share WSDL definitions, XML schema, policy expressions and so on. For the case of OPC UA, the GetEndpoints Service, which is part of the OPC UA standard, retrieves the information required to allow for secure communication between clients and servers. The information mainly consists of addressing and security policies and definitions (Dürkop et al., 2012; Ballinger et al., 2008; Mahnke et al., 2009).

Service description in Arrowhead is dependent on the method of service discovery implemented. As previously mentioned, the DNS system uses DNS-SD guidelines to organise the resource records. In such a case, the specification already allots a structure for the definition of addressing, service name, and other connection-related information. For any additional requirements, the DNS TXT record is capable of accommodating such information. The XML-based system uses XML schema for the description of data, and the Web Application Description Language (WADL), which is WSDL's counterpart for RESTful services for the description of service interfaces. The JSON-based system, as previously mentioned, is yet to be defined (Blomstedt and Olofsson, 2016b; Blomstedt, 2016a).

[2] http://www.yaml.org/.

14.4.3 Data Representation and Access

The aspect of data representation and access has been covered somewhat partially while discussing realizations of service descriptions. Our focus here is a more in-depth description of the information or data model and the semantics used by the architectures' respective implementations. The main purpose of these models and semantics is to homogenize the representation of information or data in the entire system ensuring accessibility by all participants and avoiding the need for any data transformation (Erl, 2008).

Of the five projects only two, IMC-AESOP and Arrowhead, rely primarily on mature standardized models for the representation of data. In IMC-AESOP, the system applies the OPC UA information model to link up information in the majority of the enterprise, save the lowest of layers, which instead uses a custom data model based on the Sensor Markup Language (SenML). The former, OPC UA, contains a flexible address space that can be used to create information models that capture objects, their attributes, and relationships. These objects are known as nodes in the OPC UA address space and can be used to represent physical or virtual components. The resulting information models create full-mesh networks of these nodes, with associated properties and relations, and are exposed to applications through OPC UA servers (Colombo et al., 2014b; Mahnke et al., 2009; Henßen and Schleipen, 2014).

As for SenML, the associated specification defines a data model suitable for highly constrained devices, such as sensors and actuators. It does so by having a minimalist approach where the goal is to maximize the amount of information not included in a message while still allowing for self-describing data that includes measurements and meta-data. The result is a single array data model that contains a series of data records. The records can contain the device's unique identifier, a time stamp, the measurement value and unit, amongst other details, for example, to allow for the description of the measurements and device, in addition to the measurement values themselves. The IMC-AESOP project, however, claims that the level of granularity of information carried by SenML's data model are insufficient for its needs and therefore create a custom data model that is primarily and heavily based on SenML to achieve this granularity (Jennings et al., 2016; Colombo et al., 2014b).

Similar to IMC-AESOP, Arrowhead also identifies OPC UA and SenML as suitable candidates for the implementation of data structures and semantics. However, it also highlights the Home Performance XML standard and the CoRE Link Format as appropriate for its needs. The former is a set of data standards that define a number of XML schema and associated data elements to allow for the description of customers, contractors, buildings (and their components and systems), and energy performance factors such as conservation, consumption, and savings, both as actual readings and as estimates. The goal of these standards is therefore, as the name implies, standardization in the collection and transfer of information in the domain of home performance. The latter, the CoRE Link Format, is a realization for exposing the

URIs of resources on constrained devices and networks. It does so by extending the HTTP Link Header format to include the URI descriptions, such as resource relations and attributes, as a message payload, and specifying an entry point URI as a default request path for the retrieval of these URIs. We do note, however, the existence of divergences from the semantic and modeling technologies listed in the Arrowhead guidelines as the energy production demonstrator pilot, for example, adopts the domain-specific language, Thing Markup Language (ThingML), instead of the ones listed above for its semantic needs (Varga et al., 2014; Building Performance Institute, 2013a,b; Shelby, 2012; Arrowhead Consortium, 2014b).

Opposed to IMC-AESOP and Arrowhead are the ontology-driven eScop and IoT@Work models. eScop, as previously mentioned, develops a proprietary Manufacturing System Ontology (MSO) based on OWL to describe the system components, their attributes and relationships. The MSO is in fact the evolved form of the Politecnico di Milano Production Systems Ontology (P-PSO), which is a general taxonomy for discrete manufacturing systems. The MSO extends the P-PSO to include logistics and process production from the perspective of control. The MSO also incorporates concepts to allow for the visualization of the respective systems and their data. The information is stored in an RDF triple-store database that supports SPARQL-over-HTTP to allow for web-based interactions with the ontology (Fumagalli et al., 2014; Negri et al., 2015; eScop Consortium, n.d.(c)).

The IoT@Work project also follows an ontology based approach by storing information on devices in a DS-specific data model (ontology) that uses an RDF-triple-store. In addition to RDF, the model is also inspired by the uCode Relation Model that models device profile attributes as subject-predicate-object triples. Effectively, the resulting DS model is a connected directed graph where the vertices are physical or virtual entities or primitive elements and the edges in the graph represent the relationships between the various entities and elements. The DS is also capable of validating and handling requests for information on devices acquired through an exposed RESTful interface. This information may be retrieved from the database or by collecting this information directly from connected devices. The IoT@Work-compliant devices, unlike the DS, use the OPC UA address space and information model. The project also supports the retrieval of information from SNMP compliant devices. Mappings between the respective device and DS models are therefore a necessity (Rotondi et al., 2013; Imtiaz et al., 2013).

The PLANTCockpit project presents its own metamodel for a database schema and an XML schema for the storage of visualization engine configurations. The database schema consists of four customizable data types and runtime data types to allow for the persistence of analytics-relevant data. The XML schema contains a number of elements to allow for the rendering of SVG components in HTML5 pages, and their linking to data points to create a configurable and compound HMI made up of different graphical elements (PLANTCockpit Consortium, 2012b).

14.4.4 Information and Message Encoding

For message encoding, all of the projects employ XML and extend support for one or more other specifications. XML is in fact a platform-independent data structuring format that defines rules for the textual encoding of human and machine-readable data. It allows for user-defined tags and different data types and processing methods. By allowing for the definition of syntax rules and standardized contracts through the use of document type definition (DTD) or XML schema definition (XSD) descriptions, the validation and verification of encoded data structures are possible. Several specifications have since been defined for the binary-encoding of XML documents to address the overhead and performance issues associated with XML. Of the possible choices, the Efficient XML Interchange (EXI) specification is employed by the IMC-AESOP and Arrowhead projects for the compact exchange of information (PLANTCockpit Consortium, 2012d; Hill, 2015; Varga et al., 2014; Colombo et al., 2014b; Rotondi et al., 2013; Negri et al., 2016).

Other than XML, JSON is also widely employed, being used by all but PLANTCockpit. Stipulations in the PLANTCockpit approach do however allow for the inclusion of JSON. This is the case, for example, with the configuration connector module which is required to be format-agnostic in handling configuration data. As for the JSON specification itself, JSON, like XML, is a data structuring specification that defines rules for the formatting of exchangeable and human and machine-readable data. Tools for the parsing and generation of JSON exist for a large number of programming languages therefore making it a popular alternative to XML. It follows a minimalist encoding approach, using a small number of characters to denote the structure and value of data. Similar to XML, JSON allows for the definition of JSON-based schema for the validation of resulting encodings. Binary encodings, such as the Concise Binary Object Representation (CBOR) specification, exist for JSON. However, aside from the mention of CBOR support as a long-term goal for Arrowhead's historian, it does not appear as though any of the projects include a binary encoding for JSON in their respective stacks (Colombo et al., 2014b; Rotondi et al., 2010; Varga et al., 2014; PLANTCockpit Consortium, 2011b; eScop Consortium, n.d.(b); Galiegue et al., 2013a,b; Eliasson, 2015).

Aside from XML, EXI, and JSON, three other formats used are OPC UA Binary, HTML, and XML-binary Optimized Packaging and Message Transmission Optimization Mechanism (XOP/MTOM). OPC UA Binary, as the name implies, is the binary protocol for OPC UA. Like other binary representations, it is the performance and overhead-sensitive data format for OPC UA. It is used in both IoT@Work and IMC-AESOP and is considered to be part of the PLANTCockpit OPC UA adapter. HTML, on the other hand, is used for the structuring and presentation of multimedia web content using human and machine-readable semantic descriptions. It is explicitly stated as part of the visualization layers of IoT@Work and PLANTCockpit. Finally, XOP/MTOM, is used by IMC-AESOP for the transmission and reception of binary data in SOAP messages (Rotondi et al.,

2013; PLANTCockpit Consortium, 2012a,b; Colombo et al., 2014b; Imtiaz and Jasperneite, 2013; PLANTCockpit Consortium, 2012c).

Further specifications include security relevant ones, such as the XML-based Security Assertion Markup Language (SAML) and eXtensible Access Control Markup Language (XACML), which are employed in IoT@Work, SOAP for the structuring of messages, and the previously mentioned YAML for the description of RESTful APIs. The first of these, SAML, is a standard for the communication of data for authentication and authorization, while XACML handles the definition of access policies. How these are applied as part of IoT@Work will be discussed later on in this chapter. SOAP defines a platform independent XML-based framework for message structuring, encoding, and processing and for the representation of remote procedure calls and responses. The SOAP message structure is made up of a SOAP envelope, SOAP body, and, optionally a SOAP header. The first, the SOAP envelope, is used to represent the message itself, while the SOAP header can be used to add features and their associated attributes to a message. Lastly, the SOAP body contains the message contents to be conveyed to the other communicating parties. The platform-agnostic nature of the SOAP protocol is one of the driving factors behind its popularity in the web services community. As for YAML, this was discussed earlier as the language of choice for the configuration of Swagger files under the eScop project. YAML, like its counterparts, is a serialization language aimed at minimizing the number of characters required to indicate the structure and value of data while maximizing human readability in the resulting data interchange format. It is built around a typing system, an aliasing mechanism and primitives such as mappings, scalars and sequences. According to the YAML specification, compared to JSON, it is more difficult to generate and parse but more legible to humans than JSON. The specification also states that there is no direct correlation between XML and YAML (Rotondi et al., 2013; OASIS Security Services (SAML) TC, n.d.; Iarovyi et al., 2015a; PLANTCockpit Consortium, 2012d; Ben-Kiki et al., 2009; Box et al., 2000).

14.4.5 Message Exchange

For message exchange, we note a preference for web-based solutions, such as HTTP and CoAP, and message-oriented middleware (MOM), such as JMS, AMQP, XMPP, and MQTT.

Starting with HTTP, this application-level, stateless, and generic protocol is used by all projects for the transfer of hypermedia across networks. The protocol typically runs over TCP, employs MIME-like messages for communication, and uses URIs to provide access to resources. For its secure equivalent, HTTPS, HTTP is transported over a TLS tunnel. In several instances, such as IMC-AESOP and PLANTCockpit, implementations used HTTP/HTTPS for the conveyance of SOAP messages. CoAP, on the other hand, is a low overhead URI-based web protocol for machine-to-machine (M2M) communication over UDP with support for unicasting,

multicasting, proxying, caching, stateless HTTP mapping, and binding to DTLS. Due to properties such as these, a number of projects, namely IMC-AESOP and Arrowhead, have favored the use of the CoAP protocol for the access of constrained devices and network implementations in their realizations (Fielding and Reschke, 2014; Rescorla, 2000; Varga et al., 2014; Rotondi et al., 2010; Colombo et al., 2014b; PLANTCockpit Consortium, 2012c; Severa et al., n.d.; Shelby et al., 2014; Eliasson, 2015; Derhamy, 2015).

As for the MOMs employed, PLANTCockpit uses JMS, IoT@Work employs AMQP, and certain Arrowhead demonstrators implement XMPP or MQTT. MOMs are a paradigm for asynchronous, loosely coupled and reliable communication in distributed systems. These properties, as well as others such as high scalability and availability, are enabled through the use of an intermediate layer, the middleware, that handles the messaging process on behalf of the communicating parties. The first MOM to be discussed is JMS, which is a vendor-agnostic standard that defines a Java API and semantics for the description of the interface and the messaging system behavior It therefore allows applications to communicate with heterogeneous enterprise messaging systems, and simplifies their development process. However, the implementation of the messaging service itself is not defined by the standard. As such, only a very general structure for the JMS message is defined by the standard necessitating the inclusion of integration techniques if multiple implementations of MOM exist within the same system (Mahmoud, 2004; Richards et al., 2009; PLANTCockpit Consortium, 2012a,d; Imtiaz et al., 2013; Skou et al., 2014; Eliasson, 2015; Derhamy, 2015).

In contrast to JMS, AMQP defines an open-standard messaging protocol that includes the networking protocol and message structure and remains agnostic towards the client API and message broker employed. Its protocol provides flow control features, message delivery guarantees, and highly flexible routing mechanisms for communicating parties. It appears that IoT@Work based its decision on using AMQP for its ENS implementation on the fact that it addresses, as a standard, both aspects of high-level modeling and wire-level communication concepts (OASIS, 2012; Luzuriaga et al., 2015; Richards, 2011; Imtiaz et al., 2013).

In terms of Arrowhead's choices, the XMPP and MQTT protocols are either highlighted for use or directly implemented in a number of its services and demonstrator pilots. The former, XMPP is a widely-implemented XML and TCP/IP based protocol for near-real-time communication. The protocol addresses aspects related to connection establishment and teardown, security, discovery, reliability, messaging, and inter-entity interactions. MQTT, on the other hand, is a lightweight protocol for constrained and unreliable systems that is more efficient than HTTPS, but is not extensible, and does not natively include connection security, transactions, discovery, or message fragmentation features. Multiple instances in the Arrowhead documents list the desire for services to support both MQTT and XMPP, or, in the case of the mediator service, to support translation between the two protocols (Le Pape et al., 2014; Skou et al., 2014; Eliasson, 2015; Derhamy, 2015; Gazis et al., 2015; Saint-Andre, 2011).

Other protocols noted include DPWS and OPC UA. DPWS uses SOAP-over-HTTP and SOAP-over-UDP bindings, yet certain member services, such as WS-Discovery and WS-Transfer, are transport independent. With SOAP-over-HTTP the SOAP message is placed inside the HTTP payload field for request/response messaging allowing for features from both specifications. Similarly, the SOAP-over-UDP binding allows for the inheritance of UDP's messaging, encoding, security and other mechanisms.

Regarding OPC UA, four combinations of encoding, security, and transport protocols are possible:

- UA Binary + UA-SecureConversation + UA-TCP
- UA Binary + HTTPS
- UA XML + SOAP + HTTPS
- UA XML + WS-SecureConversation + SOAP + HTTP

Of these four, the first of these compositions, referred to as native UA Binary, is mandatory for implementation. Both of IMC-AESOP and IoT@Work use the native UA Binary profile for the transport layer of their OPC UA implementations. IMC-AESOP, however, also includes a communication interface in its DPWS stack that implements the third profile made up of UA XML, SOAP and HTTPS, labeling it as OPC UA over WS. Arrowhead, on the other hand, presents a proof of concept of a condition monitoring system that employs the OPC UA SDK from Unified Automation, and discusses its use for the integration of legacy components in its framework. Finally, although OPC UA is highlighted as a development technology for PLANTCockpit, the details of the implementation is unclear from the public deliverables (Jeyaraman et al., 2008; Zurawski, 2005; Moreau et al., 2007; Colombo et al., 2014b; Mahnke et al., 2009; Bony et al., 2011; Varga et al., 2014; Le Pape et al., 2014; Hästbacka et al., 2014; PLANTCockpit Consortium, 2012c).

14.4.6 Networking, Data Link and Media

As may have been partially visible from the previous parts of this chapter, a wide and heterogeneous variety of media and their associated specifications are used, required, or discussed for possible implementations. Effectively, however, only three projects, Arrowhead, IMC-AESOP, and IoT@Work, explicitly state the technologies implemented at the networking, data link and media layers.

The Arrowhead framework's stack is declared as being made up of IPv4/IPv6, 6LoWPAN, 802.11p, 802.15.4, NFC, UWB, and NTP. However, we also note that, other than the aforementioned protocols, several more are used in one of the pilot demonstrations. Specifically, the GSM (2G), GPRS (2.5G), UMTS (3G), WiFi and RS-485 CAN standards are highlighted by the Arrowhead project as possible solutions for communication in electrical vehicle charging infrastructure. The wireless specifications are to allow users to communicate with charging stations using devices separate from the vehicle, such as mobile devices. So far,

the pilot demonstration has limited its implementation to UMTS and WiFi. As for the communication channel between the vehicle and the charging station, the Arrowhead project uses CAN, basing it on the CHAdeMO standard (Varga et al., 2014; Bellavista and Ornato, 2014; Arrowhead Consortium, 2014c; Bocchio and Ornato, 2014; Kleyko, 2016).

The IMC-AESOP project declares a stack somewhat similar to Arrowhead, using IPv4, IPv6, TCP, UDP, 6LoWPAN, IEEE 802.15.4, IEEE 802.11 and NTP. However, it uses RS-485 Modbus instead of CAN, and also includes Profibus, UA Native, and IEEE 1588 PTP. The majority of these protocols, namely IPv4, IPv6, TCP, UDP, 6LoWPAN are part of the DPWSCore stack in IMC-AESOP. The component responsible for bridging the DPWS and OPC UA stacks implements the UA Native protocol. A pilot demonstrating the migration of a plant's lubrication system to the IMC-AESOP approach used the Modbus protocol to connect to the distributed control system and a specific stack consisting of XML/EXI, CoAP, NTP, UDP, IP, 6LoWPAN, IEEE 802.15.4. A second pilot, for building system of systems with SOA technology highlights the integration of smart home systems with communication infrastructure using SenML, EXI, CoAP, IPv6, IEEE 802.11, IEEE 802.15.4 and cellphone communication technologies (agnostic) (Colombo et al., 2014b).

The IoT@Work approach necessitates the use of IPv6 and designates IPv4 as optional. For its DS, as previously discussed, both NFC and QR codes are supported. For timing, both NTP or IEEE 1588 PTP are possible. To demonstrate the auto-configuration system, it uses the RT Ethernet standard Profinet. Its network slices technology is mapped using Ethernet VLAN. Where IoT@Work differs from other approaches is its focus on discussing facilitating protocols such as SNMP, DHCP, DNS, LLDP, and STP to support the network-level autoconfiguration of devices (Rotondi et al., 2013; Houyou et al., 2011; Imtiaz et al., 2013).

14.4.7 Security

The IoT@Work approach to security is founded on capability-based access control. This mechanism centers around the use of transmissible tokens that reference an element and its access rights. A process in possession of a valid token may therefore interact with the referenced element within the constraints of its access rights. These capabilities may be forged or revoked in the form of XML documents following specific schema with elements borrowed from the SAML, XACML, Digital Signature and XML encryption schema. IoT@Work also requires that devices have secure identifiers based on the IEEE 802.1AR specification, and that authentication for NAC be carried out based on IEEE 802.1X. With these mechanisms in hand, the system designs and employs multiple components in a SO fashion to perform NAC and to secure the system's event namespace (Gusmeroli et al., 2013; Rotondi et al., 2013; Fischer and Gesner, 2012).

The PLANTCockpit project explored the possibility of using single-sign on solutions for a security service. Ultimately, the PLANTCockpit approach employs an LDAP service for access control and the management of user rights. Interactions with the LDAP service are through a Java client implemented using the JLDAP library. User access rights govern the components and data sources that a user is permitted to interact with. As for the security aspects related to PLANTCockpit's use of JMS, notes in the deliverables claim that PLANTCockpit permits the encryption of the JMS message body, with the encryption to be managed using a central component and that JMS security is addressed using 'interceptors'. Unfortunately, the security aspects of PLANTCockpit are addressed in a non-public deliverable (D3.2) and, as such, details on the interceptors and other system mechanisms for security are not available for us to elaborate further (PLANTCockpit Consortium, 2011b, 2014, 2012d).

As far as the published materials go, Arrowhead instills security measures for its Service Discovery and Authorization Control, for mediation in legacy systems, and for communication in general. The Service Discovery service is secured by using DNS TSIG keys for DNS updates and DNS Security Extensions (DNSSEC) for queries. Authorization Control is dependent on the use of X.509 certificates, and provides TLS security for the establishment of secure communication links. In the case of Arrowhead's Virtual Market of Energy pilot, the Authorization module uses Public Key Infrastructure (PKI) and X.509 certificates over REST for authentication and XMPP-over-TLS for encrypted communications. Generally, the consensus throughout the Arrowhead framework is to secure information exchange using TLS. In the case of UDP communication, DTLS is highlighted as the applicable counterpart. The project has also expressed a desire to develop a secure NFC interface for industrial applications. Based on a publication, this aspect may have been addressed through the 'ESTADO' system for smart maintenance. This system uses a 'CUT-IN' module made up of a secure and a non-secure but more powerful controller. The module is designed to be an add-on that would provide security features to 'non-smart' and legacy devices. This would include abilities for the secure storage and execution of data, integrity checks, encrypted memory and encrypted in-CPU calculations. Finally, we note the use of the Message Passing Interface (MPI) with Secure Shell (SSH) for protected communications in the implementation of a distributed framework for 3D swarming systems such as aerial vehicles and WSNs (Blomstedt and Olofsson, 2016a,b; Varga et al., 2014; Chrysoulas and Jansson, 2016; Krimm et al., 2014; Lesjak et al., 2014; Sadrollah et al., 2014).

In the case of IMC-AESOP, by self-admission, "cyber-security was not at the heart of [this] project" (Colombo et al., 2014b). As such, IMC-AESOP states that WS-Security and WS-ReliableMessaging were not implemented as part of its DPWS communication stack, and neither was IPsec included in the IPv6-based stack for WSANs, claiming them all as planned additions. Furthermore, the service bus in IMC-AESOP only implemented HTTP basic authentication and Role-Based Access Control (RBAC) for service calls with such rights only being given to administrative users. In accordance with RFC 7617, unless communication takes place within a secure system (i.e., over TLS), basic authentication is not to be

considered secure as credentials are transferred in clear-text. It is unclear if IMC-AESOP implements basic authentication over a secure channel (Colombo et al., 2014b; Reschke, 2015).

Similarly, other than security testing and a high level discussion of defining and applying a security model as part of an SOA ecosystem, eScop did not address the aspect of security. In the case of the former, security testing of the RPL is defined as a method for verifying the robustness of the ontology by employing "ad-hoc offensive queries". For the VIS layer, testing is to be applied to access control measures. As for the testing of the integration of the PHL, RPL, VIS and ORL, this entailed ensuring that data in the system is secured and that the functionality of the system cannot be misused (eScop Consortium, 2014; Simone et al., n.d.).

14.5 Discussion

The concept of SOAs has been in existence for well over a decade and is generally considered to be a stable and tried architectural design pattern. The evolution of SOAs and alternatives to the SO approach within the context of industrial systems are partially addressed in Chaps. 8 and 13. This chapter, on the other hand, inspects the architectural designs and technological choices of five preliminary SO RAs. Based on this examination, a number of features and limitations common to all five projects are apparent. One such observation is the lack of homogeneity in approach by all five projects in defining their respective architectures. Differences in the levels of abstraction, the aspects chosen for specification, and the selected forms of representation are some noted dissimilarities. For example, IMC-AESOP pursues a high level of abstraction, and therefore limits its reference architecture to a definition of service groups, member services, dependencies, and possibilities for inter-service interactions. It does not specify any protocols and standards as part of the reference architecture and instead addresses these aspects through prototypical demonstrators and pilot projects. In direct contrast are the IoT@Work and Arrowhead projects which provide concrete and detailed architectures with specific protocols, standards and specifications selected for various elements of their architectures. In terms of representation, it may be observed that only IMC-AESOP, eScop and Arrowhead declare their use of semi-formal notation in their respective documents, while IoT@Work and PLANTCockpit lack such explicit statements. Note that a related study on software reference architectures concludes that informal specification is found to introduce a lack of clarity and precision in architectures by leaving room for interpretation (Angelov et al., 2012).

Further issues include missing real-time communication features critical to industrial applications. IMC-AESOP, IoT@Work, and Arrowhead all include protocols to interface with RT systems in their demonstrators or pilot projects. They also have a number of publications that take steps towards addressing real-time aspects in processing, communication, and applications such as auto-configuration (Lindgren et al., 2013; Jammes, 2011; Pietrzak et al., 2011; Dürkop et al., 2012; Durkop et al.,

2013; Garibay-Martínez et al., 2013, 2014a,b). Yet, the services themselves, and the majority of the selected technologies, remain largely without RT-capabilities. This introduces a concern as to their actual ability to operate in safety-critical CPPS environments.

Finally, in the case of interoperability, it appears that integration is still a necessity. This is since a number of the reviewed projects, in multiple instances, have chosen to select several technologies to address certain critical features in their architectures in the pursuit of flexibility. This seems to be the case, for example, with Arrowhead's approach for discovery and IoT@Work's requirement for mappings for its novel internal data model. Determining the overall impact of the continued necessity for integration in SOAs, as well as the effect of the missing features, the selected levels of abstraction and representation is a matter than can be addressed through follow-up surveys. Such studies may aim to draw conclusions on the success of the architectures' choices by determining adoption rates and stakeholder criticisms. Unfortunately, while such information would be of great benefit to both researchers and practitioners alike, it is largely absent from literature.

In sum, it is undeniable that interoperability in the industrial domain has been marred in complexity since its conception. It therefore appears as though integration is here to stay for the foreseeable future. Yet, the SO approach, as presented in this chapter, may be used to strengthen and structure the overall environment through clearly defined and documented systems, components, boundaries and interfaces, protocols, guidelines and policies. It is therefore encouraged that SO solutions continue development in the industrial domain. This is both to address the shortcomings of existing projects and to further the pursuit of agility in enterprises in the face of turbulent technical, economic, and legal environments.

Acknowledgements This paper is supported by Technische Universität Wien research funds as part of the Doctoral College Cyber-Physical Production Systems.

References

Angelov, S., Grefen, P., et al.: A framework for analysis and design of software reference architectures. Inf. Softw. Technol. **54**(4), 417–431 (2012)

Arrowhead Consortium: Arrowhead demo T4.2. Technical report (2014a)

Arrowhead Consortium: WP 4—T4.1 Demonstrator (1st Generation): The Safe Home. Technical report (2014b)

Arrowhead Consortium: WP3.1.2 PO 002 - Requirements definition - Communication and services. Technical report, Arrowhead Consortium (2014c)

Ballinger, K., Bissett, B., et al.: Web Services Metadata Exchange 1.1 (WS- MetadataExchange). (2008)

Bellavista, P., Ornato, M.: Arrowhead D3.3 Appendix 3.3b PO 3.3-002 and PO 3.3-004 of Task 3.3 Details about First Generation Pilot Results. Technical report (2014)

Ben-Kiki, O., Evans, C., et al.: YAML Ain't Markup Language (YAML) (tm) Version 1.2., YAML.org. (2009)

Blomstedt, F.: Service Discovery REST_WS-XML-SPSDTR Version 1.1. Technical report, The Arrowhead Consortium (2016a)

Blomstedt, F.: ServiceRegistryBridge Version 1.1. Technical report, The Arrowhead Consortium (2016b)
Blomstedt, F., Olofsson, P.: Orchestration System Version 1.1. Technical report, The Arrowhead Consortium (2016a)
Blomstedt, F., Olofsson, P.: Service Discovery DNS-SD-TSIG-SPDNS Version 1.3. Technical report, The Arrowhead Consortium (2016b)
Blomstedt, F., Olofsson, P.: Service Discovery REST_WS-JSON-SPSDTR Version 1.1. Technical report, The Arrowhead Consortium (2016c)
Blomstedt, F., Contini, L., et al.: Deliverable D8.3. Technical report, Arrowhead Consortium (2014a)
Blomstedt, F., Ferreira, L., et al.: The arrowhead approach for SOA application development and documentation. In: 40th Annual Conference of the IEEE Industrial Electronics Society (IECON), pp. 2631–2637 (2014b)
Bocchio, S., Ornato, M.: Arrowhead D3.3 Appendix 3.1.1.c PO 002 of Task 3.1.1 Task and concepts. Technical report (2014)
Bony, B., Harnischfeger, M., et al.: Convergence of OPC UA and DPWS with a cross-domain data model. In: 9th IEEE International Conference on Industrial Informatics (INDIN) (2011)
Box, D., Ehnebuske, D., et al.: Simple Object Access Protocol (SOAP) 1.1. W3C Note. World Wide Web Consortium (2000)
Building Performance Institute, Inc.: BPI-2100-S-2013 Standard for Home Performance-Related Data Transfer. (2013a)
Building Performance Institute, Inc.: BPI-2200-S-2013 Standard for Home Performance-Related Data Collection. (2013b)
Bullen, G., Carter, S., et al.: Web Services Dynamic Discovery (WS-Discovery) Version 1.1. Organization for the Advancement of Structured Information Standards (2009)
Cheshire, S., Krochmal, M.: RFC 6762: Multicast DNS. Technical report (2013)
Choudri, A.: The agile enterprise. In: ReVelle, J. (ed.) Manufacturing Handbook of Best Practices: An Innovation, Productivity and Quality Focus, pp. 3–23. CRC Press, Boca Raton (2001)
Chrysoulas, C., Jansson, O.: Arrowhead System Description (SysD) - Translation System Version 0.4. Technical report, The Arrowhead Consortium (2016)
Colombo, A., Bangemann, T., Karnouskos, S.: IMC-AESOP outcomes: Paving the way to collaborative manufacturing systems. In: 12th International Conference on Industrial Informatics (INDIN) (2014a)
Colombo, A., Bangemann, T., Karnouskos, S., et al., (eds.): Industrial Cloud-Based Cyber-Physical Systems. Springer, Cham (2014b). ISBN:978-3-319-05623-4 978-3-319-05624-1
Corrêa, H.: Agile manufacturing as the 21st century strategy for improving manufacturing competitiveness. In: Gunasekaran, A. (ed.) Agile Manufacturing: The 21st Century Competitive Strategy, pp. 3–23. Elsevier Science Ltd, Oxford (2001)
Delsing, J.: Building automation systems from Internet of Things. In: Presented as an 20th IEEE International Conference on Emerging Technologies and Factory Automation Keynote Presentation, Luxembourg (2015)
Dennert, A., Montemayor, J., et al.: Advanced concepts for flexible data integration in heterogeneous production environment. In: IFAC Proceedings Volumes 46(7), 348–353 (2013)
Derhamy, H.: Arrowhead transparency protocol translation. In: Presented at the Arrowhead Budapest Meeting (2015)
Dürkop, L., Imtiaz, J., et al.: Service-oriented architecture for the autocon- figuration of real-time Ethernet systems. In: 3rd Annual Colloquium Communication in Automation (KommA) (2012)
Durkop, L., Imtiaz, J., et al.: Using OPC-UA for the autoconfiguration of real-time Ethernet systems. In: 11th International Conference on Industrial Informatics (INDIN) (2013)
Eliasson, J.: Arrowhead historian system. In: Presented at the Arrowhead Budapest Meeting (2015)
Eliasson, J., Delsing, J., et al.: A SOA-based framework for integration of intelligent rock bolts with Internet of things. In: IEEE International Conference on Industrial Technology (ICIT) (2013)
Erl, T.: SOA: Principles of Service Design. Prentice Hall, Upper Saddle River, NJ (2008)

Erl, T.: SOA Design Patterns, 1st edn. Prentice Hall Service-Oriented Computing Series from Thomas Erl. Prentice Hall, Upper Saddle River, NJ (2009)
Erl, T., Gee, C., et al.: Next Generation SOA: A Concise Introduction to Service Technology and Service-Orientation. Pearson Education, Upper Saddle River (2014)
eScop Consortium: 1st Annual Report. Technical report. http://www.escop-project.eu/wp-content/uploads/2013/05/D14_eScop_Annual_Report1_publishable-summary.pdf (2013). Visited on 08 Dec 2016
eScop Consortium: D2.4 General specification and design of eScop reference architecture. Technical report (2014)
eScop Consortium: eScop Architecture. Technical report (n.d.[a])
eScop Consortium: Orchestration Layer Training Material. Technical report (n.d.[b])
eScop Consortium: Semantic Workbench. Technical report (n.d.[c])
Faist, J., Štětina, M.: Webservice-ready configurable devices for intelligent manufacturing systems. In: IFIP International Conference on Advances in Production Management Systems (APMS). Springer, Heidelberg (2015)
Fielding, R., Reschke, J.: RFC 7235: Hypertext Transfer Protocol (HTTP/1.1): Authentication. Technical report (2014)
Fischer, K., Gesner, J.: Security architecture elements for IoT enabled automation networks. In: IEEE 17th Conference on Emerging Technologies Factory Automation (ETFA) (2012)
Fumagalli, L., Pala, S., et al.: Ontology-based modeling of manufacturing and logistics systems for a new MES architecture. In: IFIP International Conference on Advances in Production Management Systems (APMS), pp. 192–200. Springer, Heidelberg (2014)
Galiegue, F., Zyp, K., et al.: JSON Schema: core definitions and terminology. Technical report, Internet Engineering Task Force (2013a)
Galiegue, F., Zyp, K., et al.: JSON Schema: interactive and non interactive validation. Technical report, Internet Engineering Task Force (2013b)
Garibay-Martínez, R., Nelissen, G., et al.: Task partitioning and priority assignment for hard real-time distributed systems. In: 2nd International Workshop on Real-Time and Distributed Computing in Emerging Applications (2013)
Garibay-Martínez, R., Ferreira, L., et al.: Towards holistic analysis for fork-join parallel/distributed real-time tasks. In: 26th Euromicro Conference on Real-Time Systems (2014a)
Garibay-Martínez, R., Nelissen, G., et al.: On the scheduling of fork-join parallel/distributed real-time tasks. In: 9th IEEE International Symposium on Industrial Embedded Systems (SIES) (2014b)
Gazis, V., Görtz, M., et al.: A survey of technologies for the internet of things. In: 2015 International Wireless Communications and Mobile Computing Conference (IWCMC) (2015)
Gusmeroli, S., Piccione, S., et al.: A capability-based security approach to manage access control in the Internet of Things. Math. Comput. Model. **58**(5-6), 1189–1205 (2013)
Hästbacka, D., Barna, L., et al.: Device status information service architecture for condition monitoring using OPC UA. In: 19th International Conference on Emerging Technology and Factory Automation (ETFA) (2014)
Henßen, R., Schleipen, M.: Interoperability between OPC UA and AutomationML. In: Procedia CIRP, vol. 25, pp. 297–304 (2014)
Hill, B.: Evaluation of efficient XML interchange (EXI) for large datasets and as an alternative to binary JSON encodings. Technical report, DTIC Document (2015)
Houyou, A., Huth, H., et al.: D2.2- Bootstrapping Architecture. Technical report, IoT@Work Consortium (2011)
Iarovyi, S., Garcia, J., et al.: An approach for OSGi and DPWS interoperability: Bridging enterprise application with shop-floor. In: 11th International Conference on Industrial Informatics (INDIN) (2013)
Iarovyi, S., Ramis, B., et al.: Representation of manufacturing equipment and services for OKD-MES: from service descriptions to ontology. In: 13th International Conference on Industrial Informatics (INDIN) (2015a)

Iarovyi, S., Xu, X., et al.: Architecture for open, knowledge-driven manufacturing execution system. In: IFIP International Conference on Advances in Production Management Systems (APMS), vol. 460. Springer, Cham (2015b)
Iarovyi, S., Mohammed, W., et al.: Cyber-physical systems for open-knowledge-driven manufacturing execution systems. Proc. IEEE **104**(5), 1142–1154 (2016)
Imtiaz, J., Jasperneite, J.: Scalability of OPC-UA down to the chip level enables "Internet of Things". In: 11th International Conference on Industrial Informatics (INDIN), pp. 500–505 (2013)
Imtiaz, J., Dürkop, L., et al.: D. 2.5 - Integrated Secure Plug&Work Frame-work. Technical report, IoT@Work Consortium (2013)
Ismail, A., Kastner, W.: Vertical integration in industrial enterprises and distributed middleware. Int. J. Internet Protoc. Technol. **9**(2/3), 79–89 (2016)
Jammes, F.: Real time device level service-oriented architectures. In: IEEE 20th International Symposium on Industrial Electronics (ISIE) (2011)
Jennings, C., Shelby, Z., et al.: Media Types for Sensor Markup Language (SenML). Technical report (2016)
Jeyaraman, R., Modi, V., et al.: Understanding Devices Profile for Web Services, Web Services Discovery, and SOAP-over-UDP. Technical report, Microsoft Corporation (2008)
Kleyko, D.: System-of-Systems Design (SoSDD) LTU Core Framework Description. Technical report (2016)
Knapp, E.: Industrial Network Security: Securing Critical Infrastructure Networks for Smart Grid, Scada, and Other Industrial Control Systems, 2nd edn. Elsevier, Waltham, MA (2014)
Krafzig, D., Banke, K., et al.: Enterprise SOA: Service-Oriented Architecture Best Practices. The Coad Series. Prentice Hall Professional Technical Reference, Indianapolis, IN (2005)
Krämer, B.: Evolution of cyber-physical systems: a brief review. In: Applied Cyber-Physical Systems. Springer, New York (2014)
Krimm, J., Olofsson, P., et al.: Deliverable D8.2 of Work Package 8: Common Service Framework - Generation 1 Version 1.1. Technical report, The Arrow-head Consortium (2014)
Kyusakov, R., Pereira, P., et al.: EXIP: a framework for embedded Web development. ACM Trans.Web (TWEB) **8**(4), 23 (2014)
Le Pape, C., Desdouits, C., et al.: Deliverable D1.3 of WP1. Technical report, Arrowhead Consortium (2014)
Lesjak, C., Ruprechter, T., et al.: ESTADO-Enabling smart services for industrial equipment through a secured, transparent and ad-hoc data transmission online. In: IEEE 9th International Conference for Internet Technology and Secured Transactions (ICITST) (2014)
Lindgren, P., Pietrzak, P., et al.: Real-time complex event processing using concurrent reactive objects. In: IEEE International Conference on Industrial Technology (ICIT) (2013)
Luzuriaga, J., Perez, M., et al.: A comparative evaluation of AMQP and MQTT protocols over unstable and mobile networks. In: 12th Annual Conference on Consumer Communications and Networking Conference (CCNC) (2015)
Mahmoud, Q. (ed.): Middleware for Communications. Wiley, Chichester/Hoboken, NJ (2004)
Mahnke, W., Leitner, S., et al.: OPC Unified Architecture. Springer, Berlin/Heidelberg (2009)
Moreau, J., Nielsen, H., et al.: SOAP Version 1.2 Part 2: Adjuncts, 2nd edn. W3C Recommendation. World Wide Web Consortium (2007)
Nagorny, K., Colombo, A., et al.: A survey of service-based systems-of-systems manufacturing systems related to product life-cycle support and energy efficiency. In: 12th IEEE International Conference on Industrial Informatics (INDIN) (2014)
Nappey P., El Kaed, C., et al.: Migration of a legacy plant lubrication system to SOA. In: 39th Annual Conference of the IEEE Industrial Electronics Society (IECON) (2013)
Negri, E., Fumagalli, L., Macchi, M., et al.: Ontology for service- based control of production systems. In: Advances in Production Management Systems: Innovative Production Management Towards Sustainable Growth, vol. 460, pp. 484–492. Springer, Cham (2015)
Negri, E., Fumagalli, L., Garetti, M., et al.: Requirements and languages for the semantic representation of manufacturing systems. Comput. Ind. **81**, 55–66 (2016)

OASIS: Advanced Message Queuing Protocol (AMQP) Version 1.0. In: Godfrey, R., Ingham, D., et al. (ed.) (2012)
OASIS Security Services (SAML) TC: Technical report (n.d.)
Pietrzak, P., Kyusakov, R., et al.: Roadmap for SOA event processing and service execution in real-time using Timber. In: IEEE 20th International Symposium on Industrial Electronics (ISIE) (2011)
PLANTCockpit Consortium: D3.1 Initial Architectural Components of PLANTCockpit. Technical report (2011a)
PLANTCockpit Consortium: Technical Report Persistency and Synchronization Model. Technical report (2011b)
PLANTCockpit Consortium: PLANTCockpit White Paper. Technical report, http://plantcockpit.eu/fileadmin/PLANTCOCKPIT/user_upload/PLANTCockpit_D3.3_Public.pdf (2012a). Visited on 03 Mar 2015
PLANTCockpit Consortium: Data and Process Model, First Draft. Technical report (2012b)
PLANTCockpit Consortium: External Interface Specification, First Draft. Technical report (2012c)
PLANTCockpit Consortium: Generic Alarms and Events Data Format. Technical report (2012d)
PLANTCockpit Consortium: Project Final Report. Technical report (2014)
Ratovsky R., Gardiner, M., et al.: OpenAPI Specification Version 2.0. Technical report. Open API Initiative (2014)
Reschke, J.: RFC 7617: The 'Basic' HTTP Authentication Scheme. Technical report (2015)
Rescorla, E.: RFC 2818: HTTP over TLS. Technical report (2000)
Richards, M.: Understanding the differences between AMQP & JMS. In: Proceedings of the No Fluff Just Stuff ^{TM} Java Conference Series (2011)
Richards, M., Monson-Haefel, R., et al.: Java Message Service, 2nd edn. O'Reilly, Sebastopol, CA (2009)
Rotondi, D., Galipò, A., et al.: D1.1 State of the Art and Functional Requirements in Manufacturing and Automation. Technical report. IoT@Work Consortium (2010)
Rotondi, D., Piccione, S., et al.: D1.3 - Final Framework Architecture Specification. Technical report, IoT@Work Consortium (2013)
Rubio, J.: Service oriented architecture for embedded (avionics) applications. Ph.D. thesis, Technical University of Catalonia (2011)
Sadrollah, G., Barca, J., et al.: A distributed framework for supporting 3D swarming applications. In: International Conference on Computer and Information Sciences (ICCOINS) (2014)
Saint-Andre, P.: RFC 6120: Extensible Messaging and Presence Protocol (XMPP): Core. Technical report (2011)
Sauter T., Soucek, S., et al.: Vertical integration. In: Wilamowski, B., Irwin, J. (eds.) Industrial Communication Systems, 2nd edn., pp. 1–12. CRC Press, London (2011)
Severa, O., Faist, J., et al.: eScopRTU with Service Manager. Technical report, eScop Consortium (n.d.)
Shelby, Z.: RFC 6690: Constrained RESTful Environments (CoRE) Link Format. Technical report (2012)
Shelby Z., Hartke, K., et al.: RFC 7252: The Constrained Application Protocol (CoAP). Technical report (2014)
Simone, P., Obluska, I., et al.: D6.1 Test strategy. Technical report (n.d.)
Skou, A., Lino Ferreira, L., et al.: Deliverable D5.3 of WP 5. Technical report, Arrowhead Consortium (2014)
Valipour M., AmirZafari, B., et al.: A brief survey of software architecture concepts and service oriented architecture. In: 2nd International Conference on Computer Science and Information Technology (ICCSIT) (2009)
Varga, P., Martínez de Soria, I., et al.: Deliverable D7.3 of WP 7. Technical report, Arrowhead Consortium (2014)
Zurawski, R. (ed.): The Industrial Communication Technology Handbook. Industrial Information Technology Series. Taylor & Francis/CRC Press, Boca Raton (2005)