



Document title: D10.3 Standardisation report year 2

Version
1.0

Author
Michel Iñigo, MONDRAGON S.Coop

Status
Final

Contact
minigo@mondragoncorporation.com

Date
2021-04-30

Deliverable D10.3 Standardisation report year 2

Work package leader: Michel Iñigo
minigo@mondragoncorporation.com

Abstract

This document constitutes deliverable D10.3 Standardisation report year 2 of the Arrowhead Tools project .



ECSEL EU project 826452 - Arrowhead Tools
Project Coordinator: Professor Jerker Delsing | Luleå University of Technology

Table of contents

1. Executive Summary	7
1.1 Standardisation Contacts and Involvement	8
2. Introduction	10
2.1 Objectives and Scope	10
2.2 Outstanding outcomes from D10.1 Standardisation base line	11
2.3 Outstanding outcomes from D10.2 Standardisation report year 1	13
3. Reference model and methodology standards	16
3.1 Smart Manufacturing Reference Models	17
3.1.1 RAMI 4.0	17
3.1.2 Asset Administration Shell	19
3.1.3 Industrial Internet Reference Architecture (IIRA)	20
3.1.4 ISO/DIS 23247-1 Digital Twin framework for manufacturing	21
3.1.5 IEC/CD TR 63319 A meta-modelling analysis approach to smart manufacturing reference models	22
3.2 Critical review of SMRMs and Gap analysis of the Engineering Process Model for managing a digitalised life-cycle of products	24
3.2.1 Gaps and barriers in SMRMs	25
3.2.2 Engineering Process for Software Development	29
3.2.3 Engineering Process in the manufacturing Industry	30
3.3 Industrial experiments considering the gaps of SMRMs and Engineering process	32
3.3.1 Industrial use case using Asset Administration Shell.	32
3.3.2 The practical implications of standardization used in the SOA context.	37
4. Language standards	40
4.1 Language standards in Engineering Industrial Systems	40
4.2 SysML and UML	41
4.2.1 The Arrowhead Framework profile for SysML V1	41
4.2.2 The Arrowhead Validation use case on Service-Oriented Architecture in SysMLv2	46
4.3 The Arrowhead Tools experiment on SysML v2 API.	50
4.3.1 Background on logical modelling tools	51
4.3.2 SysMLV2 introduction	53
4.3.3 SysMLV2 API model overview	56
4.3.4 Consuming the SysMLV2 API: example of output	59
4.3.5 Evaluation: status and plan	60

4.4 MOTIF Model transformation & integration	61
4.5 Express/STEP/ISO 10303.....	62
4.5.1 Feasibility study for a geometry/topology ontology	63
4.5.2 ISO 10303 Extended Architecture	64
4.5.3 Harmonized terms and vocabulary for industrial data	66
4.5.4 Use of Reference Data in ISO 10303	66
4.6 OWL and RDF – languages for ontologies	67
4.6.1 Standard Semantic technologies	67
4.6.2 Standardization efforts in domain knowledge representation.....	70
4.6.3 Our contribution in the standardization of domain knowledge representation.....	71
4.7 Overview of Industry 4.0 Solutions Based on Semantic Web Technologies	72
4.7.1 Ontologies for Manufacturing Management Systems.....	72
4.7.2 Ontologies for Industry 4.0 Standards	75
4.7.3 Ontology-based manufacturing management systems	76
5. Interface and industrial communication standards	77
5.1 Industrial protocols and characteristics	79
5.2 Communication Protocols Dimensions within the Arrowhead Framework	79
5.2.1 Introduction	79
5.2.2 Communication Protocols' Potential Dimensions	81
5.2.3 Communication Protocols Dimensions covered by AHT framework	89
5.2.4 Analysis of some Industrial Communication Protocol Complementing AHT framework	90
5.3 Other potential interface and industrial communication standards working through Arrowhead Tools Project	95
6. Appendixes.....	97
7. References.....	98
8. Conclusions	105
9. Revision history.....	108
9.1 Contributing and reviewing partners	108
9.2 Amendments	108
9.3 Quality assurance.....	108
10. Appendix 1 – List of standards and communication protocols.....	109

List of Figures

Figure 1: Technological Gap Between Current Manufacturing System and Industry 4.0 [1].....	16
Figure 2: Reference Architecture by ISO IEC 42010:2011	17
Figure 3: Reference Architecture Model Industrie 4.0 (RAMI 4.0) [4].....	18
Figure 4: Administration shell for digitalization [9].....	19
Figure 5: The functional viewpoint of IIRA [4];	21
Figure 6: A Digital Twin Reference Architecture Model [22]	26
Figure 7: The IBM Industry 4.0 Architecture [26]	27
Figure 8: Six propositions considering the barriers and lacks of SMRMs [25].....	28
Figure 9: Possible Sub-Models of Asset Administration Shell- ZVEI	29
Figure 10: Architecture of the use case	33
Figure 11: Components of the Robotic Arm.....	33
Figure 12: Snapshot of an OPCUA client connected to the Robotic Arm AAS.....	35
Figure 13: Grinding Machine’s AAS	35
Figure 14: Asset semantic representation example using turtle	37
Figure 15: Toolchains reference demo architecture.....	38
Figure 16: Stereotypes SD, SD-DD and IDD	42
Figure 17: System and Cloud Stereotypes	43
Figure 18: Arrowhead adaptor and a sequence diagram of how OPC-UA	44
Figure 19: Arrowhead library in SysML v2 corresponding to Arrowhead profile	48
Figure 20: The Norwegian use-case from Productive4.0 in Arrowhead notation.	49
Figure 21: Mock-up prototyping of an Arrowhead DSL description	49
Figure 22: Sequence diagram with inline nested ports	50
Figure 23: SysMLV2 API models (source: SysMLV2 specification [47]).....	56
Figure 24: SysMLV2 API PSM based o the OpenAPI specification (source: endpoint provided by Intercax).....	56
Figure 25: Project API operations (source: SysMLV2 official documentation [47]).....	57
Figure 26: Configuration management (versioning) API operations (source: SysMLV2 official documentation [47]).....	57
Figure 27: Element and configuration management (versioning) API model (source: SysMLV2 official documentation [47]).....	57
Figure 28: Element and relationship API model (source: SysMLV2 official documentation [47]).	58
Figure 28 Element and relationship model (source: SysMLV2 official documentation).	58
Figure 30: Element and relationship API operations (source: SysMLV2 official documentation [47]).	58
Figure 31: Query API model (source: SysMLV2 official documentation [47]).	59

Figure 32: Query API operations (source: SysMLV2 official documentation [47].	59
Figure 33: ISO 10303 STEP document structure	63
Figure 34: Representation of the principles of the modular STEP information model architecture ..	65
Figure 35: Semantic Web technology stack	68
Figure 36: RDF graph example	69
Figure 37: RDFS inference example	69
Figure 38: Knowledge exchange with a global ontology	71
Figure 39: Methodology Process	78
Figure 40: QoS architecture within the Arrowhead framework (Source: [93], [94])	78
Figure 41: OSI model	80
Figure 42: TCP/IP model	80
Figure 43: Star topology	82
Figure 44: Mesh topology	82
Figure 45: Bus topology	83
Figure 46: Ring topology	83
Figure 47: Tree topology.[98]	83

List of Tables

Table 1: Standardisation Contacts and Involvement.....	9
Table 2: Summary of engineering processes, methods, techniques and tools for UC3.	51
Table 3: Some outputs of the main operations offered by a SysMLV2.....	60
Table 4: QoS aspects covered by the Arrowhead Framework based on [93], [94].....	79

1. Executive Summary

The present document reports the standardisation activities associated with WP10 along year 2 of Arrowhead Tools project. The involvement and interests of the consortium in standardization and the requirements of each use case area a result of year 1 of the project embodied in “D10.1 Standardisation base line” and “D10.2 Standardisation report effort year 1”. Current activities have been focused on the contributions in the areas of Language Standards (T10.1), Reference Model and Methodology Standards and Interface (T10.2) and Industrial Communication Standards (Task 10.3) been part of WP10.

Considering that the Arrowhead Tools Project aims to develop cost and time-based on more effective engineering tools as well as interoperability among heterogenous systems, the standardisation analysis done in D10.1 and D10.2 and the standardization contributions carried out in the present deliverable are crucial to carry out the digital inclusion.

Furthermore, as a reminder, the contributions done are not focused on creating and driving new standards but intends to try and influence standards and frameworks that are of particular interest to the project and its members, reflected in each task of WP10.

For all that considerations the document remarks three different issues associated with each task in WP 10:

- **Language Standards (Task 10.1)**

It considers standards for description of systems with models defined in standard modelling languages. In this regard contributions of UML SysML V1 and V2 and ISO 10303 are considered and described. Furthermore, the inclusion of SysML within Arrowhead Tool Framework as well as an experiment for validating SysML V2 is described together with SysMLV2 characteristics. Finally, MOTIF, OWL¹, RDF² and Ontologies are also explained.

- **Reference Model and Methodology Standards (Task 10.2)**

Smart Manufacturing Reference Models (SMRMs) such as RAMI 4.0, The Industrial Internet Reference Architecture (IIRA), Asset Administration Shell (AAS), ”Digital Twin form Manufacturing” and” A Meta-modelling analysis approach to Smart Manufacturing Reference Models” are analysed in order to know their principal characteristics, standards as well as gaps and barriers. IoT, 5G, Digital Twin, Service Oriented Architecture (SOA) and industrial implementations are the principal gaps of SMRMs which should consider the Engineering Process as Arrowhead Tools Project establishes.

¹ <https://www.w3.org/TR/owl-ref/>

² <https://www.w3.org/RDF/>

Implementation of industrial use case considering Asset Administration Shell as well as Tool Chain demo are described and deployed.

- **Interface and Industrial Communication Standards. (Task 10.3)**

An analysis of the 83 Industrial Communication Protocols is listed together with characteristics. Then OPC-UA, MQTT and UMATI as key industrial communication protocols are analysed as well as their principal dimensions for knowing which of them are offered by Arrowhead Tools Framework. Furthermore, apart from an experiment for proving that in the next period NFC and SenML has been taking into account for contributions.

1.1 Standardisation Contacts and Involvement

Participant short name	Standardisation Contact Name	Email
JOTNE	Kjell Bengtsson Jochen.Haenisch	kjell.bengtsson@jotne.com Jochen.Haenisch@jotne.com
HIOF	Øystein Haugen	oystein.haugen@hiof.no
IncQuery	Géza Kulcsár	geza.kulcsar@incquerylabs.com
CEA	Saadia DHOUIB	saadia.dhouib@cea.fr
UC3M	José María Alvarez Eduardo Cibrian	joalvare@inf.uc3m.es
MGEP	Felix Larrinaga	flarrinaga@mondragon.edu
FAUT	Carlos Yurre	yurre@otek.es
IKERLAN	Cristobal Arellano Fernando Eizaguirre	carellano@ikerlan.es feizaguirre@ikerlan.es
FARR	Jon Rodriguez Mikel Viguera	j.rodriguez@fagorarrasate.com Mikel.viguera@fagorarrasate.com
CISC	Christian Ettinger	c.ettinger@cisc.at
MSI	Peter Craamer	pcraamer@msigrupo.com
REUSE	Roy Mendieta	roy.mendieta@reusecompany.com
INFINEON	Germar Schneider	Germar.Schneider@infineon.com
POLITO	Gianvito Urgese	gianvito.urgese@polito.it

Participant short name	Standardisation Contact Name	Email
EUROTECH	Azzoni, Paolo	paolo.azzoni@eurotech.com
UNIBO	Federico Montori	federico.montori2@unibo.it
DAC	Marek Tatara	marek.tatara@dac.digital
REPLY	Nigro Gerry	g.nigro@reply.it
BEIA	Crstina Istrate	cristiana.istrate@beia.ro
MON	Carolina Mejia	cmejia@mondragoncorporation.com
MON	Michel Iñigo	minigo@mondragoncorporation.com

Table 1: Standardisation Contacts and Involvement

2. Introduction

2.1 Objectives and Scope

The deliverable “10.3 Standardisation report effort year 2” is fully related to the activities under the Arrowhead Tools “WP 10 Standardisation” in response to Phase II. The deliverable addresses the challenge of Phase II with the inclusion of the contributions in the areas of Language Standards, Reference Model and Methodology Standards, and Interface and Industrial Communication Standards considering the outstanding gaps analysed in Phase I answering the challenges of T10.1, T10,2 and T10.3.

Thereby, the D10.1 shows a whole view of the standards which Arrowhead Tools members are interested or part in Standardisation Development Organizations (SDOs), Working Groups or Task Forces, the D10.2 is the overview of accompanied standards for each Use Case, and the current D10.3 accomplishes the contributions of the standardization interesting areas. The next steps to deploy, as part of Phase III, will focus on finishing the standardization contributions as well as setting up a standardization Guide and ROI tool even though is not included as part of WP 10 objective. As a result, industrial partners could address the challenge of the standardization and measure the impact of the standards in terms of engineering cost, productivity and efficiency considering an easy roadmap. Besides, it can help for the digital inclusion. That outcome would be based on the work previously developed in the deliverables and added value for all work done before.

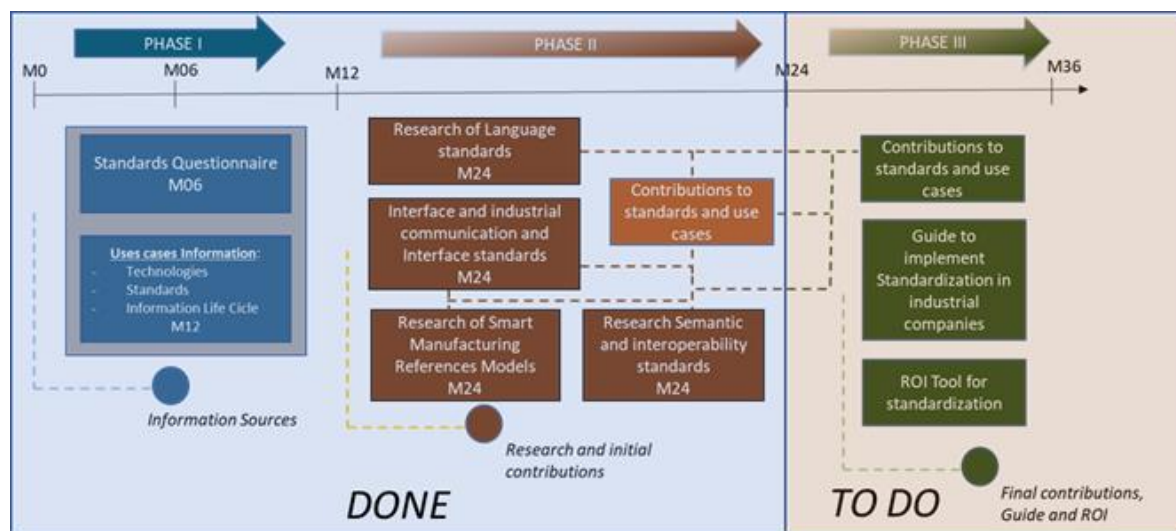


Figure 1: WP10 Workplan

As the Arrowhead Tools project does not intend to create and drive new standards but pretends to try and influence standards and frameworks that are of particular interest to the project and its members the current deliverable becomes especially relevant. Phase II has also considered the standardization contributions as key role for use cases development. Therefore, each task has considered an industrial demonstration to validate the usefulness of the standardization contribution. The Tasks 10.1 and 10.3 will deploy an industrial demonstration in within the last year of the Arrowhead project. The present deliverable shows the industrial analysis and application of Task 10.2.

The goal of the deliverable is to provide standardization contributions in the areas of Language Standards, Reference Model and Methodology Standards, and Interface and Industrial Communication Standards. The contributions carried out are essential for manufacturing and engineering area as well as the development of the use cases based on automation system life cycle management.

To achieve the main objective above, the following specific objectives will be addressed:

- Evaluate the gaps of Engineering Process and Smart Manufacturing Reference Models such as RAMI and IIRA for having a comprehensive approach developing a tool chain based on Arrowhead Tool Framework. The Asset Administration Shell is also analysed based on the gaps that SMRMs dispose.
- Describe the advantages and new functionalities in relation to Language Standards mainly focused on SysML-V2 and ISO 10303 – STEP as well as an experiment providing those contributions. MOTIF, OWL, RDF and Ontologies are also considered.
- Enumerate the outstanding industrial communication protocols together with their principal characteristics and dimension properties. Those dimensions are compared with the dimensions that Arrowhead Tools Framework offer. An industrial demonstration will be also considered to develop along Phase III.

2.2 Outstanding outcomes from D10.1 Standardisation base line

The contextualisation of standardisation in current manufacturing processes and its relevance in digital transformation exploring the main challenges industrial companies must approach while implementing structured standards. The interconnected industries should consider three kinds of integrations; End-to-end integration, Horizontal integration, and Vertical integration with enough flexibility to maximise efficiency. The main standards identified to the industry 4.0 by the principles standardisation stakeholders are the ones of communication protocols, interface and data exchange, semantic, interoperability, management and software frameworks.

On the other hand, the smart manufacturing reference architectures and models that support a company in setting up its entire production based on a jointly agreed standard solution, such as RAMI 4.0, Smart Manufacturing ecosystem developed by NIST and IIRA. Furthermore, the document describes the most relevant Standards Setting Organisations (SDOs) and alliances.

Each Arrowhead Tools partner received a standardisation survey receiving a response of 36 of them, as a significant sample of the consortium gathered for the D10.1. The partner survey main objective was to identify the standardisation involvement and interest in a particular standard or standards group, and how each organization relate to the specific standard or group. The roles were: Charing/co-chairing, actively contributing, member (rather monitoring or observing), member on national level, user, or interested.

A major target point for Arrowhead Tools partners is to automate more – on the factory floor, throughout the supply chain and during the maintenance during the lifespan of the products.

The results obtained from the partners' inputs reflect the existing need in industrial environments in terms of interoperability in the transmission of data as well as its format and suitable representation between the different OT-IT layers. Interoperability for the use of IoT / SoS Engineering solutions to favour the digital transformation that allows meeting the global objectives of the Arrowhead Tools project and standardisation. With the partners' inputs receiving a response of 37 institutions' interests related to standardisation, seven major groups of standardisation areas have been identified:

- ***System and Software***

The current requirements of industrial environments that compel the management of industrial control systems as an asset within the upper layers of the automation pyramid through IT software solutions, such as *ISO / IEC 42010 Systems and software engineering*, *ISO 15288 Systems and software engineering - System life cycle processes* or *IEC 62890 Life Cycle Management*.

- ***Information and Representation***

The standard format with which data is transmitted and the representation of the properties of heterogeneous industrial devices and systems, becoming a real need to agilely comply not only with the much-desired interoperability, but also for the representation of digital twins and management of their life cycle. One example could be the *ISO 10303 (STEP) Industrial automation systems and integration - Product data representation and Exchange*.

- ***Semantics and Language***

The representation and knowledge associated with specific applications and domains require the use of languages prepared for it. The principal standards are related to W3C and OMG.

- ***Communication***

Partners are users or are interested in interoperable protocols and which can be used in different layers of an industrial environment such as OPC-UA (IEC 62541), the recent UMATI standard for machine tools or the well-known OneM2M, MQTT.

- ***Reference Model***

The need to standardize industrial processes and their representation throughout their life cycle and in the automation pyramid through one of the standards previously seen in the areas of Communication, Semantics, etc, as seen in RAMI 4.0 and IIRA reference architecture and models.

- ***Cybersecurity and Safety***

The partners are aware of Cybersecurity, not only at the IT level with the *ISO / IEC 27001* standard, but also at the OT level with the *IEC 62443*. Nor can't ignore the interest in Safety in industrial environments as equally or more important than at the software level.

Finally highlight the awareness of authentication and trust for contactless devices, web applications, or for AI.

- ***Domain-Specific Standard***

It should be noted the active participation of partners in standards associated with Industry 4.0 technologies and therefore facilitators for the introduction of solutions and engineering tools in the working committees of Robotics (*ISO / TC 299*), Artificial intelligence (*ISO / IEC JTC 1 / SC 42*), Internet of Things, Blockchain and Digital Twin. Nor can't forget specific regulations for the Oil & Gas sector, semiconductors or standards associated with Energy and environmental management.

2.3 Outstanding outcomes from D10.2 Standardisation report year 1

The deliverable 10.2 “Standardisation report effort year 1” was focused on the requirements that are part of a specific use case intended to be displayed in Phase I of WP10. A major target point for Arrowhead Tools is digitalisation and automation solutions for the European industry, which will close the gaps that hinder the IT/OT integration.

From the responses of the Use Case template based on “IEC 62559-2:2915 Use case methodology”, and information from Milestone I, 146 standards were identified for 22 used cases and sub-cases. The results obtained from the partners' feedback reflect a strong necessity to propose a general approach in terms of interoperability and integration raw data, pre-processed information, the uniformity of the data format and the important role of the semantics as well as communication. The following four sections are the most relevant considerations to highlight:

- **Data Interchange, Semantic and Systems Modelling**

The most common and claimed necessity of the use cases and sub-use cases involved in D10.1 is the STD 90 - JavaScript Object Notation (JSON) Data Interchange Format and Extensible Markup Language (XML). The format data as well as representation and knowledge associated with specific applications and domains requires the use of languages prepared for it. From semantic perspective and implementation is essential for the interoperability of the data since it contains its meaning. Semantics play an important role in communication between machines in particular, since machines often cannot suppose these relationships from the context of the information. Furthermore, during the product life cycle, information will need to be exchanged in a way that can be understood by all partners, and JSON and XML will be the case for a 63,3% of the participating use-cases. Likewise, UML, SysML and RDF are representative standards consider for Systems Modelling. Furthermore, although standards such as ISO 10303 or RDF which were not mentioned by use cases, they were highlighted to be part of this deliverable contributions due to importance in modelling and life cycle.

- **Communication Protocols**

The communication matters are also relevant for the participant use cases. Partners are interested in interoperable protocols that can be used in diverse layers of the industrial environment, highlighting OPC-UA, MQTT, Modbus-TCP, Z-Wave, HTTP, Websocket. The security, reliability, and interoperability of transporting raw and pre-processed data are valuable for the Arrowhead Tools partners. The organization based on layers has the main advantage to allow interchangeability between their implementations, assuring technological independence via wireless or cabled. The protocols can be used as uniform interfaces to access information from machines from diverse manufacturers, simplifying the integration of components and plants, increasing the efficiency. In order to avoid redundant data generation data losses and set up intermediate layers, the standards and protocols highlighted above by partners are relevant for the use cases. Other protocols that are appropriate to point out are UMATI, CAN-Open, SigFox, Lora among others although less representative.

- **Engineering Process Model and Smart Manufacturing Reference Model**

Use cases showed interest to overcome their own challenges in the management of the development of the engineering tools associated with ISO/IEC/IEEE 15288:2015 Systems and Software engineering — System life cycle processes and ISO/IEC 12207 - Information Technology / Software Life Cycle Processes.

On the other hand, although the standardisation approach was not the principal priority of the use cases at the beginning of Arrowhead Tools project since there are few cases considering (AS-IS) that issue, their subsequent standardisation analysis has showed the relevance for carrying out the use cases taking into account the standardisation approach (TO-BE) together with technological challenge as part of the development cost and time-based on more effective engineering tools. Therefore, the global approach of standardisation focuses on Smart Manufacturing Reference Model such as RAMI 4.0 or Asset Administration Shell initiative was considered.

3. Reference model and methodology standards

This section is related to the Task 10.2 and the principal aim is to monitor the Smart Manufacturing Reference Models for providing contributions in this regard. One of the principal needs for Industrial companies is to have a common and uniform way to implement the Industry 4.0 approach. As consequence although SMRMs satisfy it, nowadays it covers only from the whole perspective it consequently becomes necessary to carry out real uses case implementations covering firstly the principles of SMRMs.

Secondly, the Arrowhead Tools Framework and its engineering process as ISO/IEC 81346 proposes should be considered since it enables the flexibility of the developments to implement not only automation perspective but also the digitalization perspective using advanced Engineering tools covering the whole life cycle. In the following figure analysed in [1], [2] and [3] we can see the gap from different Manufacturing System which Flexibility and Standardization are some of the main principals concepts to satisfy. The digitalization perspective should be aligned with the Flexibility principal in their manufacturing systems.

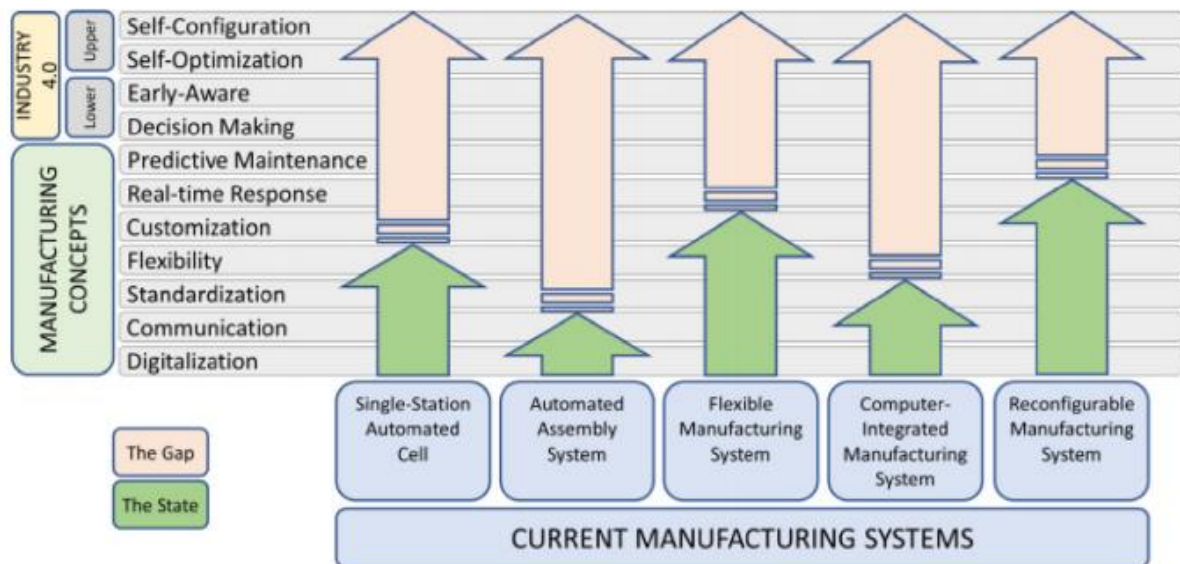


Figure 1: Technological Gap Between Current Manufacturing System and Industry 4.0 [1]

This section describes the principal characteristics of SMRMs such as RAMI 4.0 together with Asset Administration Shell, Industrial Internet Reference Architecture (IIRA) and ISO/DIS 23247-1 Digital Twin framework for manufacturing. Contributions considering gaps of SMRMs are described in this regard:

- An analysis of the needs and gaps of SMRMs considering the perspective of the Engineering Process and the implementation of digitalization, interoperability and Integration IT/OT together with automation.

- A contribution considering the Asset Administration Shell with the implementation of the Identification and Monitorization sub-models of two industrial devices.
- The need of Service-orientation as the main enabler of value network integration and collaboration as well as smart plug-and-produce shop-floor systems. Toolchain reference demo was developed to show how standardization is useful in practice.

3.1 Smart Manufacturing Reference Models

“ISO IEC 42010:2011 Systems and software engineering — Architecture description” illustrates a reference architecture as the structure of a system with its element types and their structures as well as their interaction types among each other and with their environment. A Reference Architecture defines restrictions for an installation (concrete architecture). Through abstraction from individual details, a Reference Architecture is universally valid within a specific domain. Further architectures with the same functional requirements can be constructed based on the reference architecture.

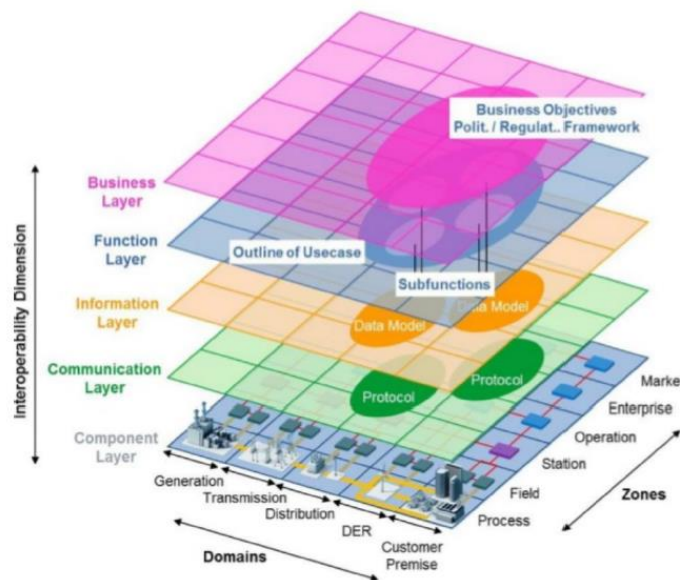


Figure 2: Reference Architecture by ISO IEC 42010:2011

3.1.1 RAMI 4.0

The Reference Architecture for Industrie 4.0 [4] (RAMI 4.0) is the result of cooperation between experts from Technical Committees 7.21, “Plattform Industrie 4.0”, and 7.20, “Cyber-Physical Systems” of the VDI/VDE Society for Measurement and Automatic Control (GMA). Have made a major contribution to the results considering interests meet in the

discussion concerning Plattform Industrie 4.0 initiative ³ which involves industries from process to factory automation with totally different standards, information and communication technologies and automatic control with associations of Bitkom, VDMA, ZVEI and VDI, and the standardization organizations IEC and ISO with their national mirror committees in DKE and DIN.

The RAMI 4.0 is an adaptation and expansion from the Smart Grid Architecture Model (SGAM) in the pursuit of unifying procedures for working in industrial environments serving as basis for discussion of its relationships and details in Industry4.0 [5]. The RAMI 4.0 consists of a three-dimensional model to represent the I4.0. The corresponding axes of the model are listed below:

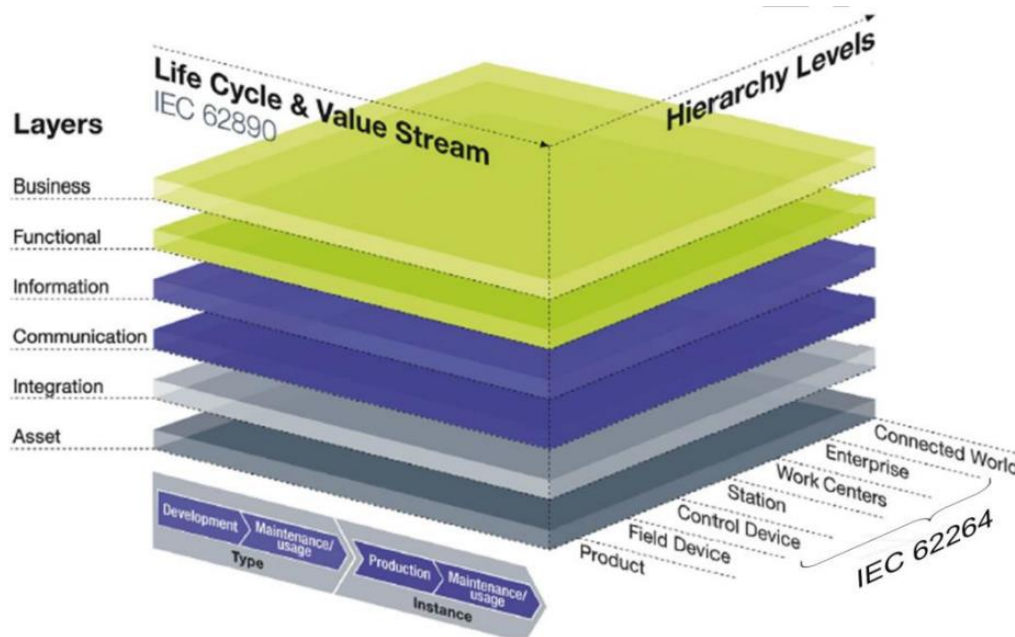


Figure 3: Reference Architecture Model Industrie 4.0 (RAMI 4.0) [4]

The Layers axis consists of layers of asset, integration, communication, information, functional, business.

- **The Asset layer** represents a physical facility, a device, or a product.
- **The Integration layer** represents the administration shell, which allows the digital transformation of a physical asset adding a software layer considering standardization meta-model for transforming it as a digital item [6]. Once it has been converted into a digital asset, the above layers of RAMI4.0 expands toward communication, information, functional, and business layers which can utilize the digitalized asset. Further information related to the Administration Shell can be seen in section 3.1.2.
- **Business layer:** Functions that map business models/processes, define rules and regulations, and orchestrate services.

³ <https://www.plattform-i40.de/>

- **Functional layer:** Functions that formally describe, model, and integrate services.
- **Information layer:** Functions that support event pre-processing, execution of rules, data analysis, and quality assurance.
- **Communication layer:** Functions that support communication and provide control services.

The **lifecycle and value stream axis** is based on "IEC 62890 Life-cycle management for systems and products" used in industrial process measurement, control and automation which standardizes lifecycles of automotive production plants or chemical plants. It consists of stages of type and instance. The type stage is divided into development and maintenance/usage. The instance stage is divided into production and maintenance/usage. The axis of lifecycle and value stream.

Finally, **the Hierarchy axis** levels consists of product, field device, station, work centers, enterprise, and connected world. It shows expansion in space along the hierarchy of physical plant facilities. Further information per each area can be seen in [7]. The axis of hierarchy levels is a variant of "IEC 62264 Enterprise-control system integration" or "IEC 61512 Batch Control".

3.1.2 Asset Administration Shell

An Administration Shell or Asset Administration Shell (AAS) is a "standardized digital representation of an asset such as an industrial physical asset, software etc. This concept is the corner stone of the interoperability between the applications managing the manufacturing systems or Cyber Physical Systems. AAS identifies the Administration Shell and the assets represented by it, holds digital models of various aspects defined as sub models and describes technical functionality exposed by the Administration Shell or respective assets" [8]. Furthermore, it turns an object or asset into an intelligent component for Industry 4.0 (I 4.0 component). It is based on IEC/TS 62832 Digital Factory standard, and defined in a draft mode in IEC 63278 which does not only provide the standardization and the digitalization of an asset, but also the interoperability among them.

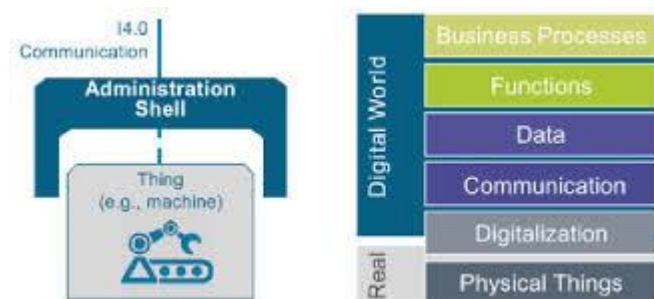


Figure 4: Administration shell for digitalization [9]

In Figure 4, an example of an I4.0 Component, shows the integration of an asset (Machine) with its Administration Shell illustrating that the administration shell is the element where the communication layer, information layer, functional layer and business layer are deployed based on RAMI 4.0 [10], [11]. Finally according to [12], AAS is meant to be implemented for its representation in the following technology such as XML + services, API REST + JSON, OPCUA.

An Industry 4.0 system can be described as the integration of I4.0 Components, which have the following properties[13] :

- The components of the same level can interact and cooperate with each other to meet common and individual goals.
- Each component may be composed by other components of a lower level and be a part of a higher level component.
- All have the same fundamental internal structure but with different roles.

Digital Twin can be represented as Asset Administration Shell. In this context in [14] discusses the terms “Asset Administration Shell” and “Digital Twin” and classify them in relation to the plant life cycle 4.0. The authors conclude that the Asset Administration can be treated as a Digital Twin considering a fully enriched version. As a result, AAS enables to add not only a basis standardization representation of an asset but also representation of key standardized properties of life-cycle product and manufacturing. Furthermore, considering the geometry and kinematics properties as well as sequential behaviour and signals, the Asset Administration Shell can be defined through XML representation described in AutomationML .COLLADA and PLCOpen format can also be added considering the same representation of AAS [14] . The initiative Industrie 4.0 from Germany has also established the concept of AAS as a Digital Twin in [15].

Finally and regarding semantic perspective, in [16] proposes a Semantic I4.0 component using RDF for data interoperability, HTTP Unique Resource Identifiers (URIs) for global unique identification of the assets, data base SPARQL for querying the data, translations of existing standards into RDF vocabularies and semantic web technologies to facilitate multilingualism using AAS. Moreover, in [12] developed ”both a landscape of Industry 4.0 related standards and the Standard Ontology (STO) for the semantic description of standards and their relations” .They populated the ontology with standards such as RAMI or NIST and important concepts for the domain such as the Administration Shell submodel.

3.1.3 Industrial Internet Reference Architecture (IIRA)

The Industrial Internet Reference Architecture (IIRA) [17] is a standardized open architecture based on industrial production systems developed by the US-led Industrial Internet Consortium (IIC), The main scope of IIRA is to maximize its value of broad industry applicability to drive interoperability, map eligible technologies, and technological

guidelines and standard development. Furthermore, it is globally the main driver behind worldwide adoption of Industrial Internet of Things (IIoT).

The IIRA abstracts the common characteristics, features and patterns from use cases define in communication, energy, healthcare, manufacturing, security, transporting and logistics domain, that have been defined by the Industrial Internet Consortium (IIC). IIRA establishes four viewpoints: Business, Usage, Functional and implementation.

- **The Business viewpoint** addresses business-oriented concerns such as how the system delivers value to the business and how it aligns with business strategy as well as financial concerns such as expected Return on Investment (ROI).
- **The Usage viewpoint** takes the business requirements and realizes them through creation of user and system activities that deliver the required outcomes and business objectives.
- **The Functional viewpoint** addresses the stakeholders' concerns regarding the functionality of the Industrial Internet system.
- **The Implementation viewpoint** deals with the technologies needed to implement functional components (functional viewpoint), their communication schemes and their lifecycle procedures.

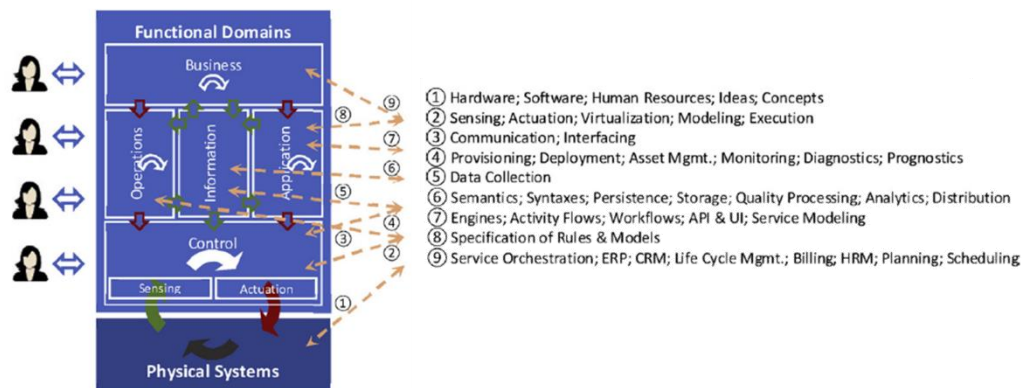


Figure 5: The functional viewpoint of IIRA [4];

Finally, in [18] demonstrates the mapping between the IIRA 3-tier functional viewpoint with the IT layers associated with the RAMI 4.0 architecture for the interconnected industrial organization and systematic model for asset efficiency testbeds.

3.1.4 ISO/DIS 23247-1 Digital Twin framework for manufacturing

The ISO 23247 series of standards is called "Automation systems and integration — Digital Twin framework for manufacturing" and is currently being developed by ISO/TC 184/SC 4/WG 15, "Digital Manufacturing".

ISO 23247 currently consists of four documents, which recently passed their DIS-ballot as a package. The information in this chapter is based on these DIS-documents. All four documents are currently under ballot resolution and final editing before their publication. Though the DIS ballot resolutions should not change the technical contents significantly, this chapter should be revised after publication of ISO 23247.

ISO 23247 is described to be a framework to support the creation of digital twins of observable manufacturing elements. Its main use case is the design of architectures for digital twins of observable manufacturing elements including personnel, equipment, materials, manufacturing processes, facilities, environment, products, and supporting documents.

The Digital Manufacturing Platforms (DMP) Cluster (<https://www.efpf.org/event/DMP-Cluster-Meeting>) develops under the auspices of European Factory of the Future Research Association (EFFRA) an overview of standards that are relevant for smart manufacturing and digital twins (https://cloud.effra.eu/index.php/s/sX3CGEiYrCTUW0W?path=%2FWGs%2FWG1_Standardisation%2FT1.2%20Common%20standards). The overview includes a mapping of the standards to the RAMI 4.0 framework and its categories Resources, Communication, Information, Functional and Business. Here, ISO 23247 is identified to support the following categories:

- Communication,
- Information and
- Functional.

ISO 23247 consists of the following four parts:

- ISO 23247-1: Overview and general principles.
- ISO 23247-2: Reference architecture.
- ISO 23247-3: Digital representation of manufacturing elements.
- ISO 23247-4: Information exchange.

These documents describe the world of discourse of digital twins for manufacturing by defining terms and reference models in Parts 1 and 2, and functional, information and networking views in high level architectures in Parts 2, 3 and 4. Information requirements for the observable manufacturing elements are stated down to the attribute level. For "equipment", for example, the attribute "status" is described by enumeration values for the current state of the equipment.

Standards that may be used to represent such values, such as, ISO 10303 STEP, are listed in an informative annex of Part 3. Implementation options for the entire framework of ISO 23247 are included as an informative annex to Part 4

3.1.5 IEC/CD TR 63319 A meta-modelling analysis approach to smart manufacturing reference models

IEC 63319 is developed by JWG 21, “Smart Manufacturing Reference Model(s) (SMRMs)” of ISO/TC 184 and IEC/TC 65. The document is currently at ballot stage 30.99, that is, the Closing Disclosure (CD) is approved. As this is a Technical Report (TR), it is now being prepared for publication.

This chapter is based on the document that was used for the ballot as Technical Report: ”ISO-TC184_N1856_ISOIEC_DTR_63319_A_meta-modelling_analysis_approach_to_smart_manufacturing_reference_models.PDF”.

The title of IEC 63319 is “A meta-modelling analysis approach to smart manufacturing reference models”.

The document is a result of a collaboration of IEC and ISO recognizing “the changing dynamics of manufacturing and the potential opportunities and benefits of developing a reference model for smart manufacturing for both the developers and the users of standards.”

The objectives of a smart manufacturing reference model are:

- Guidance and support for the development and use of standards for smart manufacturing;
- Guidance and support for the development of smart manufacturing systems.

The ultimate goal of the activity is to publish an IS that specifies an “Unified reference model for smart manufacturing (URMSM)”. Thus, the document at hand is the result of a preparational phase or, as the title says, of an analysis. Commonalities among the following ten smart manufacturing reference models are identified in the draft TR:

- Scandinavian Smart Manufacturing Model
- RAMI 4.0
- IMSA
- ISO 15704 – GERAM Annex
- NIST Smart Manufacturing Standards Landscape
- KSTEP cube framework
- IVRA Next
- IIC Industrial Internet Reference Architecture
- Smart Manufacturing Standards Map (SM2)
- URM-MM.

These meta models were compared along a set of modelling concepts, such as domain, stakeholder and viewpoint, and along relationships or propositions among these concepts. Among others, the following issues and challenges were uncovered to create an implementable and successful unified smart manufacturing reference model:

- Model content purpose.
- Content Scope.
- Need for modelling languages.

- SMRM Interface to modelling languages.
- SMRM approach on implementation models.
- Approach on SW interfaces.
- Application in real manufacturing systems.
- Use in smart manufacturing operation.

A new work item proposal ballot for the follow-up project (IEC 65815) was already successfully completed on 2020-08-14 for the following scope (Smart_Manufacturing_65-815-NP.pdf):

“This document specifies a unified reference model for smart manufacturing (URMSM). This model comprises a set of common smart manufacturing modelling elements, their associations and relationship criteria. This model establishes a reference model for documenting relationships among entities involved in smart manufacturing. The reference model accommodates systems consisting of equipment, products, and services, within the domain of manufacturing. The reference model is suitable for use in new manufacturing opportunities and challenges, supporting the development of industry and country specific standards and specifications.” Also this project is managed within JWG 21.

3.2 Critical review of SMRMs and Gap analysis of the Engineering Process Model for managing a digitalised life-cycle of products

The current needs in production are related to the implementation not only agile processes but also due to customization and smart products as well as the interoperability and integration of the industrial devices and applications between production and business layer considering common procedures. Furthermore, the integration of Internet as Internet of Things, Cyber Physical Systems together with Industry 4.0 technologies is needed to satisfy the competitiveness of the industrial companies.

SMRMs have emerged to respond the needs in Manufacturing sector combining the IT technologies and standards in communication (OPC-UA), security and safety (IEC 62443, IEC 61511), Classification and product description (eCl@ss), Life Cycle (IEC 62890), Enterprise-control system integration (IEC 62264), W3C Semantic Web, Engineering needs as AutomationML etc providing common procedures.

Common shared key characteristics between RAMI 4.0, IIRA include [19]:

1. Dissolution of the automation pyramid
2. A communication solution that makes data available to all parties in real time
3. The addition of a dimension that captures the lifecycle of the product and production facility
4. Assets such as products and production resources have cyber counterparts.

On the other hand, the gap analysis discussed by [20] in the industry and software engineering domains, several Engineering Process (EP) models have been developed and used to describe the life-cycle of product/solution (P/S) and services produced.

In general, an EP is described as a workflow: a sequential description of phases and activities, during which documents, information and tasks are passed from one phase to another for action, according to a set of procedural standardised rules [21]. EPs can have branching points that result in the execution of parallel tasks and decision-making points that can guide the execution path toward many alternatives. Currently, the trend is to define 3D reference models that integrate aspects such as factory hierarchy and business architecture layers with the life-cycle value stream. A short review of EPs in software and production industries offers a proper background for a multi-dimensional solution space.

However, due to several gaps in SMRMs and EP considering the current requirements combining digitalization and automation along life-cycle product and development of engineering tools has been described in the following sections.

3.2.1 Gaps and barriers in SMRMs

The principal gaps barriers a of SMRMs are enumerated as following. It has been considered RAMI 4.0 as the most representative although IIRA, NIST model and others are also considered:

- Although SMRMs represent the cornerstone for the development of projects in industry 4.0 as RAMI 4.0, they are only focused on the definition of rules for the implementation of I4.0 applications on a **high level point of view**, and knowing that each manufacturing system requires a different, special development according to their specific requirements [13].
- There are **few implementations** which consider the standards or architecture that SMRMs purpose. Some of them use system architecture but do not integrate the standards such as in where they propose the use of AutomationML and OPC UA as future work. For more information reviews [13] paper.
- In [7] describes the limitations of RAMI 4.0 since it **uses only IEC standards** although a joint working group 5 between the IEC/ISO SC65E and the ISO TC184/SC5 technical subcommittees is actively developing a multi-part IEC 62264 standard based on the ISA-95 specifications. Furthermore, it highlights that the **relationship between two space axes** of Layers and Level is **not clear**.

As a consequence, [7] proposes to include ISO standards, the relationship between two space axes of Layer and Level and **the need of inclusion of the digital twin concept** in RAMI-4.0 in addition to the administration shell concept.

In [22] proposes a Digital Twin Reference Architecture Model in Industry 4.0 based on RAMI 4.0.

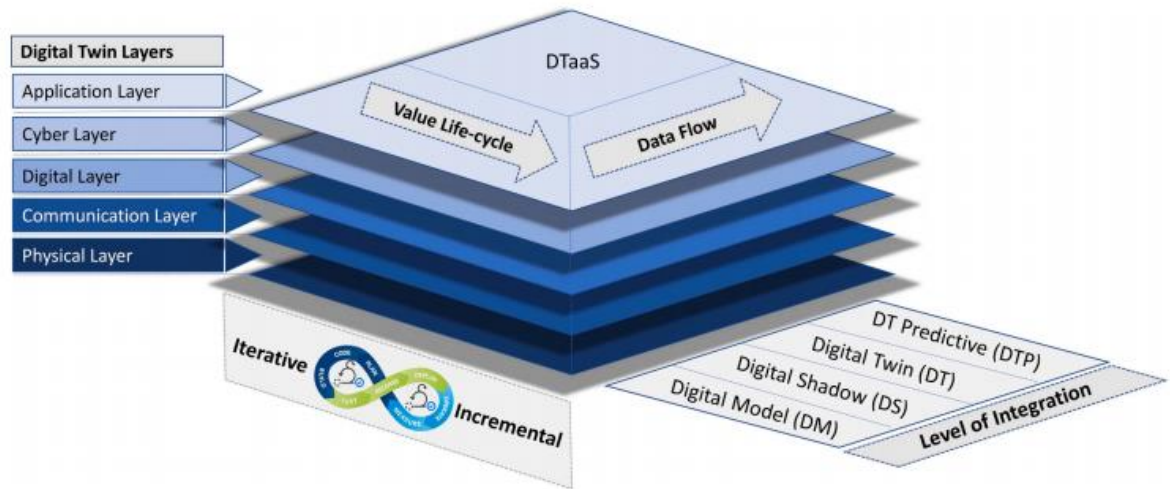


Figure 6: A Digital Twin Reference Architecture Model [22]

It suggests that the **lifecycle of a production plant** or a product can be **better represented by** adopting models from other standards such as the FIATECH model [23] or **ISO 10303** because they accommodate more stages than IEC 62890 and also covers broader kinds of product categories including a factory.

Finally, IEC 62890 standardizes lifecycles of automotive production plants or chemical plants. It focuses on the maintenance of a plant where automation devices are used for the plant operation. Because the life span of a plant is much longer than that of an automation device, upgrades of devices are expected during the lifecycle of the plant itself and this characteristic needs to be accommodated in an SMRM.

- Due to the fact that the **IoT and 5G** Telecommunication is getting more important in the smart manufacturing, RAMI 4.0 should accommodate this trend as Korean Smart Reference Model [24] in axis of telecommunication physical hierarchy.
- The functional hierarchy axle exposes in RAMI 4.0 and IIRA models, is a valid taxonomy of manufacturing functions, but it is not a sufficient functional architecture [25]. On the one hand, the required capabilities such as value network collaboration through digitalization and sharing of manufacturing resources on the IoT as micro-services the ones proposed by IBM Industry 4.0 Architecture [26] and NIST, on the other hand, suggest that the functional architecture in the context of smart manufacturing is indeed a **service-oriented architecture (SOA)**.

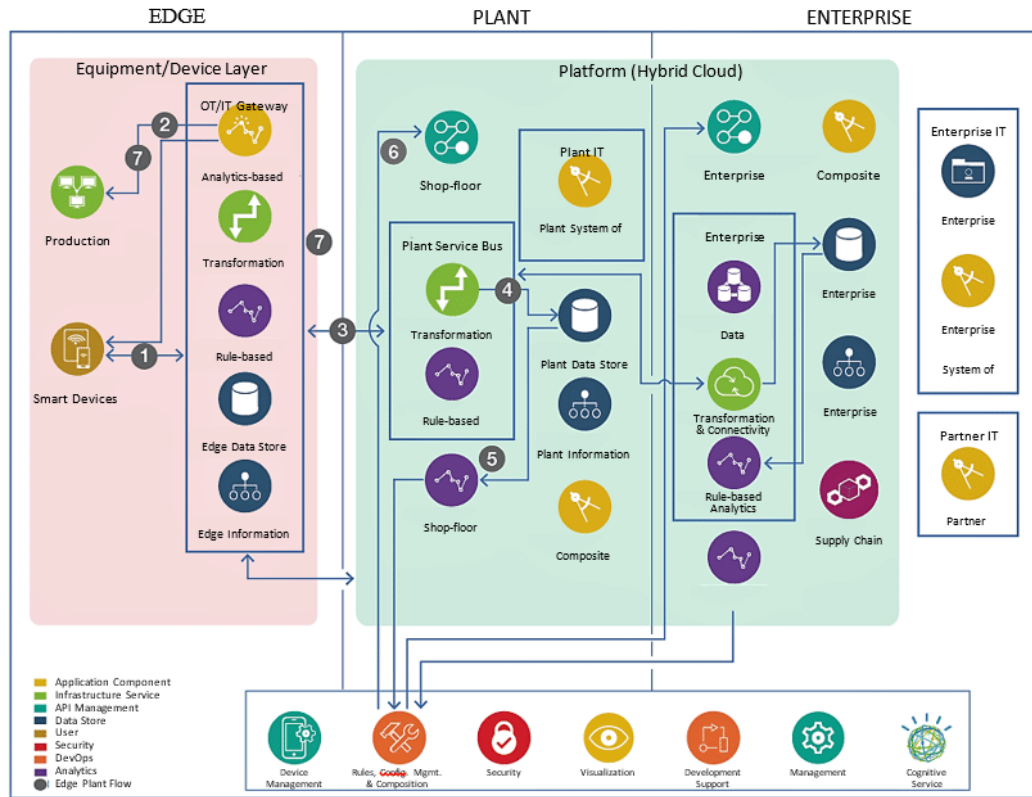


Figure 7: The IBM Industry 4.0 Architecture [26]

- In [25], considering the barriers and lacks of SMRMs, have proposed in terms of six propositions on the characteristics of smart manufacturing which should be researched for being introduced as part of them. The list that can be seen in Figure 8 which was proposed by seven experts from industry, government, and standards organizations.

Service-orientation was identified as the main theme of the proposals: digitalization and integration of manufacturing resources on the IoT as on-demand services, as the main enabler of value network integration and collaboration as well as smart plug-and-produce shop-floor systems. Therefore, [25] identifies two high-priority actionable plans:

- Modelling and composition of **micro-services**, and
- Optimizing the topology of interactions between micro-services, smart objects, and humans

The identification of those propositions as lack of Service Oriented in Smart Manufacturing Reference Models are also identified in the Section 3.2.3 considering Engineering Process in Manufacturing.

R&D Challenges	Expert						
	1	2	3	4	5	6	7
Integration of shop floor, business processes, and cloud services for plug-and-produce work (P1)			✓	✓	✓	✓	
Fully-integrated industry—inter-enterprise, cloud-based publication and sharing of services (P1)						✓	
Service representation, composition, discovery, registration, and matching (P1)			✓				
Service definition—shift of focus from macro to micro services in manufacturing SOA (P1)							✓
Representation of complex capabilities such as cognitive systems or analytics as services (P1)		✓					
Standardized taxonomy of manufacturing processes as services (P1/P2)		✓	✓			✓	
Asynchronous and interoperable communication mechanisms and publish/subscribe API (P1)	✓			✓	✓	✓	
Need for 'smarter' objects to execute more complex services (P2)							✓
Standardized description of objects as prerequisite for standardization of services (P2/P4)			✓			✓	
Common language (e.g., OPC-UA) for standardizing object and services (P2)	✓		✓		✓	✓	
Mechanisms for orchestration, filtering, aggregation, and sharing of cloud services (P3/P4)		✓			✓	✓	
New models for the economics of acquisition, implementation, and integration (P3)						✓	
New models for automated, on-the-fly creation and governance of value networks (P4)				✓		✓	
Horizontal integration of product life-cycles through micro-services (P4)		✓					
Need for autonomous, self-/environment-aware, intelligent devices for service-orientation (P4)			✓				
Concurrent optimization of order-to-cash and design-manufacture-maintenance cycles (P4)							✓
Lack of horizontal integration and reconfigurable manufacturing capabilities in ISA-95 (P5)		✓					
Integration of ISA-95 into a cloud-based architecture (P5)						✓	
Transition from thousands of enterprise-wide applications to consistent cloud-based services (P6)		✓					
Lack of modularity of legacy manufacturing systems hindering 'composability' (P6)		✓					

Figure 8: Six propositions considering the barriers and lacks of SMRMs [25]

- Considering Asset Administration Shell it is still at embryonic stage. On the one hand, several reports have defined the specification, details and structure by Industrie 4.0 initiative. Furthermore, the relationship between RAMI 4.0, Digital Twin and I4.0 components have also been defined. A guide of the Asset Administration Shell can be seen in [27].

On the other hand, there are some applications to carry out the model of assets considering Asset Administration Shell specification. The most important one is AASX package explorer which the website <https://github.com/admin-shell-io> contains the latest version and can be used to create, edit and view AAS file serializations (*.aasx). The site also includes the AASX-server which makes AASX packages accessible via REST, OPC UA and MQTT, a highly recommended FAQ with best practices and further resources.

Furthermore, BaSyx project (<https://bit.ly/3mNRe85>) provides various modules to cover a broad scope of I4.0 (including AAS). Hence its substantially more complex architecture. PyI40AAS (<https://bit.ly/307hPmJ>) is a Python module for manipulating and validating AAS. On <http://www.i40-aas.de/> one can access numerous AAS examples of different vendors based on the use case of a digital nameplate. Likewise, University of Catania and through CoreAAS (<https://github.com/OPCUAUniCT>) and OPCFoundation together with VDMA and ZVEI brought I4AAS-OPC-UA the AAS information model is enabled to set up using OPC-UA. Uninova in Portugal has also defined the tool NOVAAS.

However, even though several reports have been published, due to the complexity of the standardization ecosystem there are still no real scenarios implemented for the industry considering Asset Administration Shell. Besides, in spite of the fact that the specification, details and structure have been defined, there is a lack of standardized sub models. In [8] is described the possible sub-models of Asset Administration Shell

and the standards that are representative of each sub model, but they should be also implemented considering the structure of AAS definition. Even though, it needs further development and consensus. The following image visualizes the possible Sub-Models of Asset Administration Shell.

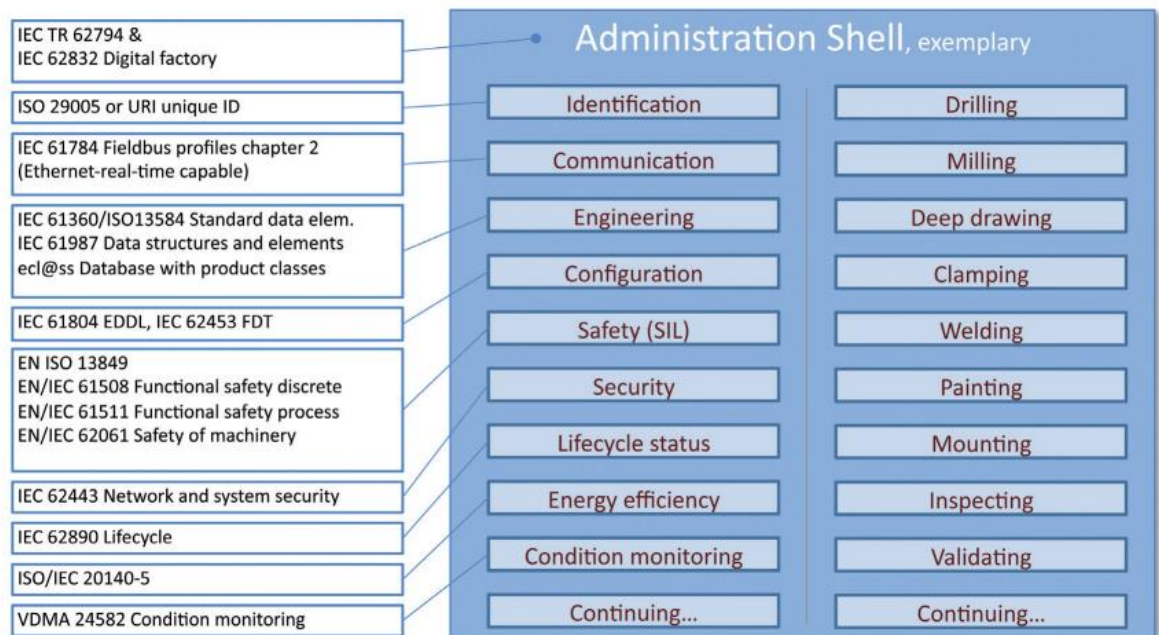


Figure 9: Possible Sub-Models of Asset Administration Shell- ZVEI

3.2.2 Engineering Process for Software Development

One of the first engineering process models, introduced by Royce in 1970 [28], is the Waterfall model, which describes the steps to be implemented in a sorted list of consequential phases. Each phase of the development proceeds in sequential order without any overlap. The result of the phase must be passed to the next phase in complete form, tested, and well documented after a predefined time period allocated for its development. Under the Waterfall model, the implementation of corrections of any defect found later on in the development of the product life-cycle is inefficient. However, the sequential nature promotes proper documentation for each phase to ensure formal information transfer between phases.

The Waterfall life-cycle model can still be used in low-complexity projects with relatively smaller development and maintenance teams. The V-model was introduced in the software engineering domain to improve some of the bottlenecks of the Waterfall model [29]. This structure allows developers and testers to work in parallel, with benefits for development time and costs.

In the V-model, relationships between each phase of the development life-cycle and its associated phase of testing are explicit. This feature ensure that the result of each phase is properly checked and approved before moving forward to the next phase. Unlike the Waterfall, the V-Model involves tester teams in the requirement phase itself. It allows a certain level of flexibility since requirement changes are possible in any phase and can be satisfied with a small overhead [30]. The V-Model is mostly used in large companies as it requires a large number of resources to support reviews and updates of each development and associated testing stage.

The Agile model [31] aims to abstract the model and documentation of the product to be developed. This model does not have a fixed structure and can be customised for different application domains and products, making the integration of values, principles, and practices more flexible in the life-cycle description of a product. The Agile methodology requires an adaptive team that is able to respond to changing requirement even late in development. Working software is delivered frequently so that customers are satisfied by the rapid and continuous delivery of new software versions.

Comparison of these three life-cycle models reveals that unfortunately, there is not an easy recipe that can be applied to all UCs. Indeed, depending on the features of the project, one needs to choose which life-cycle model is best fit for purpose [32]. However, by analysing these frequently adopted EPs, it is noticed that the process is fixed and static, presenting difficulties to correct the requirements when changes are presented. Small projects can be managed in a more flexible way with respect to large project. Therefore, ideally, it might be more convenient to partition a large project in several smaller parts, which can be developed in parallel by different teams (or different stakeholders) and integrated at the end to assemble the main product or service. This approach is particularly suited for the life-cycle management of the SoS, which can be partitioned into a collection of task-oriented or dedicated systems that pool their resources and capabilities together to create a more complex system that offers more functionalities than the sum of the single constituent systems [33].

3.2.3 Engineering Process in the manufacturing Industry

The current state-of-the-art for engineering of a production process is based on the ISA 95 architecture [34] and engineering standards such as “ISO/IEC 81346-1:2009, Industrial systems, installations and equipment and industrial products Structuring principles and reference designations”, CAEX [35],[36], “ISO 15926-13:2018 Industrial automation systems and integration — Integration of life-cycle data for process plants including oil and gas production facilities” and “IEC 62424:2016 Representation of process control engineering”.

This has resulted in highly functional but very stiff and inflexible manufacturing automation architectures and solutions. Yet, reliable flexibility has been difficult to provide in manufacturing up to now, due to the lack of engineering models capable to manage complex

use cases based on heavily interconnected components from multi-stakeholders with different EPs (e.g. IoT).

Industry 4.0 digitalisation foresees the integration of stakeholders in ecosystems, e.g., a factory, an airport, or a bridge. Such integration requires data sharing among different local automation systems owned by different legal bodies, which are possibly located in different countries under their legal systems. The sharing of data enables the optimisation of productivity, raw material yield, energy and environmental footprint, etc., which is the basic motivation for automation. Several reference architecture models are under development to improve the digitalisation level in the intelligent manufacturing domain by combining concepts, methodologies and technologies taken from the IoT, cyber-physical systems (CPSs), cloud computing, big data analytics (BDA), and information and communications technology (ICT)[37].

The new reference architectures RAMI 4.0, Smart Manufacturing Ecosystem by NIST and the IBM Industry Architecture lack the ability to address some of the relevant key points that are useful for the Service Oriented Architecture (SOA) and System of System (SoS) domains. In [25] recently interviewed a pool of experts, with the main aim of producing a critical review of the direction taken from the reference architectures designed to support the Industry 4.0 development and highlight the shortfalls of these models. In the following list, we report the major observations of this study coupled with the needs that we identified by analysing all the mentioned production engineering process models:

- None of the models expose the EP's resources as fine-grade services; all are limited providing only high-level macro services or even do not adopt an SOA model at all.
- RAMI 4.0 and IIRA are based on a slightly improved automation pyramid (IEC62264) that still presents challenges in terms of migration from a legacy control system to an SOA control system [38].
- The models do not specify how loosely coupled services, associated with EP resources, can be composed, shared, and utilized on-demand and throughout value networks of collaborating and competing stakeholders for supporting SoS UCs.
- Mechanisms for dynamic and decentralised mapping of EP resources onto micro-services have not been clearly addressed by any of the current reference architectures.
- None of the models and architectures reviewed specify how humans interact with emerging manufacturing systems and environments (i.e., human-machine symbiosis). We have to realise that this new technology requires training of people with new skills related to the understanding of the new platform, new tools and new architecture.
- None of these models support continuous engineering, making the interaction of different engineering processes from many stakeholders difficult.

- These models are not designed to explicitly support value chain and supply chain needed in the SoS domain.
- These models are mostly focused on the business-to-business UCs.
- Predictive maintenance is a significant enabler toward Industry 4.0. However, until now, it has not been considered within the framework of RAMI 4.0 to yield a unified predictive maintenance platform [39].
- None of the models address the bridging of legacy automation engineering and the new and emerging IoT and SoS platforms such that the integrated solutions can meet basic industrial requirements in terms of, e.g., real time, robustness, scalability, security, safety, and engineering simplicity. Most importantly, all these models were conceived for a specific sector without considering the partnerships/alliances between the stakeholders involved in the product value chain and required, in the current industry panorama, to address the complexity that characterises products based on IoT and SoS technologies. Thus, to successfully develop those kinds of projects, companies need interdisciplinary teams of people with heterogeneous backgrounds, collaborating and interacting through integrated and automated EP based on ICT solutions.

3.3 Industrial experiments considering the gaps of SMRMs and Engineering process

3.3.1 Industrial use case using Asset Administration Shell.

Automation is evolving from a hierarchical model towards an integrated network of smart automation devices. In this scenario, it is essential to develop interoperability tools for integrating assets in the Industry 4.0 network [26]. To overcome such integration challenges, AAS can be used for automatic self-conducted machine data exchange and for interaction and integration with the industrial environment. In order to promote standardization and interoperability from smart factory perspective, MONDRAGON corporation materialized digital transformation along its Research Technological Organizations (RTO) partners (Ikerlan, Mondragon University and Ideko) not only through R&D projects but also through the next Asset Administration Shell Use Case in Arrowhead Tools projects.

The use case presented represents an implementation of RAMI 4.0 as one of the gaps that Smart Manufacturing Reference Models need. The use case as can be seen in the following figure creates two AASs such as Robotic Arm AAS and Grinding Machine AAS and tests the interoperability between them using a semantic integrator. To improve its deployment, Docker⁴ has been used.

⁴ <https://www.docker.com>

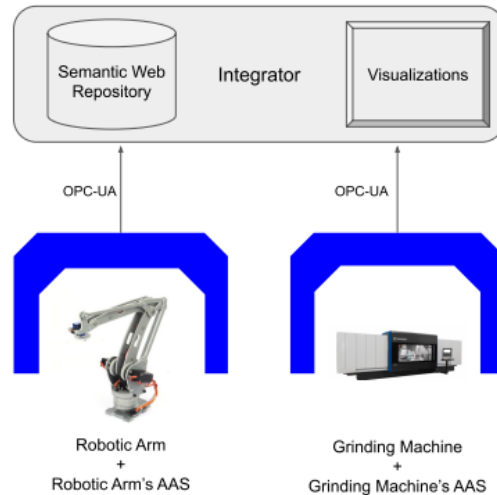


Figure 10: Architecture of the use case

a) Robotic Arm's AAS

An AAS has been implemented over a RoboticArm demonstrator. This demonstrator is based on a SainSmart Robotic Arm, a Raspberry Pi and a controller board for the Adafruit servos. The robotic arm (see Figure 11) has 4 axes controlled by four servos. The function of the Raspberry Pi is to control the movement, through a standard keyboard connected by USB, and to serve the information model of the RoboticArm. The keystrokes are transformed into the appropriate signals for the servos through the Adafruit Servo Hat card. This small board is connected above the Raspberry Pi and adds the necessary interfaces to control the servos via pulse-width modulation (PWM) signals.

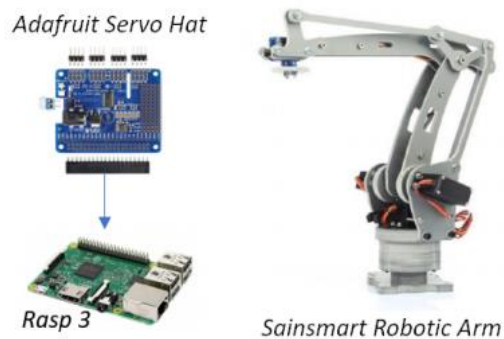


Figure 11: Components of the Robotic Arm.

Initially this part of the demonstrator (developed by IKERLAN) contained an OPCUA server that published the OPCUA Model Companion-Specification for Robotics as defined in OPC 40010-1 - Robotics Part 1⁵. This specification provides information about asset configuration

⁵ OPC 40010-1 - Robotics Part 1: Vertical Integration. OPCUA Information Model release, , <https://opcfoundation.org/developertools/specifications-opc-ua-information-models/opc-unified-architecture-forrobotics/>

and condition monitoring. It also presents the model for a motion device type and its axis type component.

This model facilitates interoperability in scenarios where communication with different types of robots is necessary, making it easier for applications to monitor or act on them. AAS goes further and proposes a more generic model in which aspects of life cycle as well as semantic content will be incorporated. The demonstrator has been transformed to consider the RoboticArm as an Asset and the OPCUA server in the Raspberry Pi has been modified to publish the Administration Shell of the RoboticArm asset instance, making use of an open source OPCUA implementation (open62541). **Identification, Documentation and Condition Monitoring have been defined as sub-models** for the Asset RoboticArm:

- The sub-model Identification contains all properties related to the identification of the asset used in the demonstrator (manufacturer, model, serial number).
- The Documentation sub-model contains information about the files that document the asset (datasheet, maintenance manual).
- The submodel Condition Monitoring contains some of the properties defined in the AxisType and that are relevant for our demonstrator such as the motion profile and the actual position of each axis of the RoboticArm.

Although the metamodel defines the concept of Operation, CoreAAS does not have it defined among its types, however its implementation of the AAS PropertyType allows not only reading but also writing the attribute Value. Thus, to act on the RoboticArm, instead of Operations new Properties have been added to the sub-model in order to move the axes. Axis X SetPosition, Axis Y SetPosition and Axis Z SetPosition Value will be set and the RoboticArm will move according to that value.

Furthermore, each element of these submodels have a semantic Id defined, using custom URIs, with the aim that the final orchestrator can interact with the different properties of the submodel. Figure 4 shows a generic OPCUA client connected to the RoboticArm AAS.

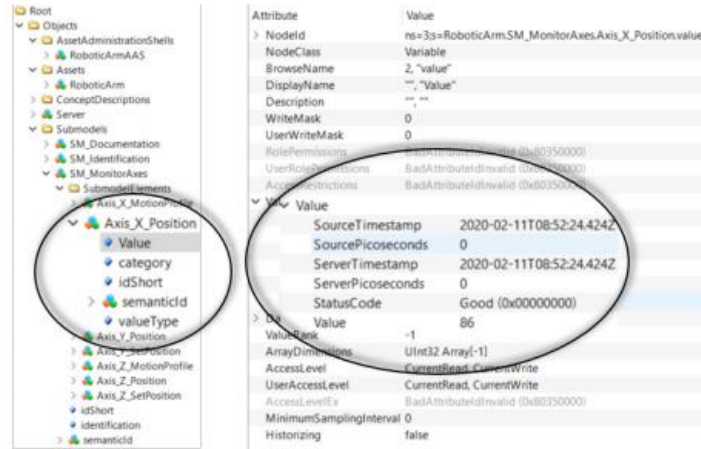


Figure 12: Snapshot of an OPCUA client connected to the Robotic Arm AAS

b) Grinding Machine's AAS

The AAS has been implemented over DANOBAT's HG72 grinding machine. The idea behind is to make the AAS implementation easily exportable to other grinding machines or even different kind of machine tools. The developed architecture is shown on Figure 5.

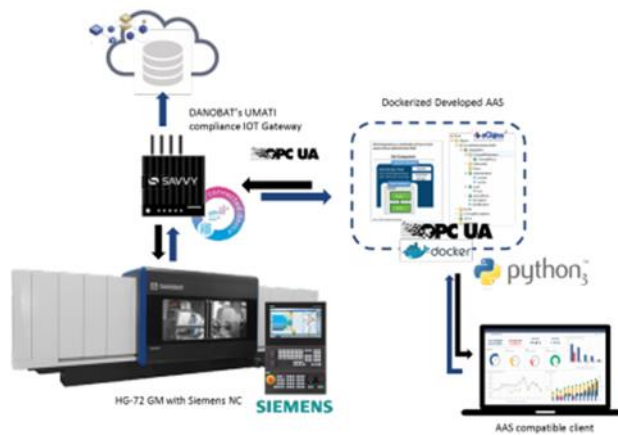


Figure 13: Grinding Machine's AAS

The communication with the machine is done using DANOBAT's data system solution and its IOT gateway, called 'SAVVY BOX'. This smart box acts also as a IT/OT gateway that transforms the custom field-bus protocols from different manufacturers (such us Siemens S7, Modbus, OPCDA, EherCAT...) onto well known IT data exchange protocols such as API REST or OPCUA. Recently, a new feature has been added that allows the box to exchange information using UMATI (Universal Machine Tool Interface), a interface that standardizes the way the machine tools share information over OPCUA ⁶.

⁶ <https://dw.de/en/technology-and-standardisation/umati-universalmachine-tool-interface/>

Using the UMATI interface, the data can be easily extracted from the machine and in a standard manner for all UMATI compliant machines. To build the AAS, an abstraction layer has been placed above UMATI, using python programming language. This software gathers the data coming from the UMATI interface and adds all semantic information to build the AAS according to the document [8] and exposes it as an OPCUA server. The semantic information has been added using custom URIs for uniquely identifying each one of the elements. Part of the model implemented in OPCUA can be seen in the next bullet points:

- CustomName: Machine's name according to catalogue.
- JobStatus: An integer that indicates the status according to UMATI standard.
- Manufacturer: The name of the manufacturer (in this case Danobat).
- Absolute Positions: X, Y and Z absolute positions of the machine.

c) Integrator

The integrator works as the plant organizer. It manages the manufacturing workflow and the communication with the different assets. It consists of 2 parts:

- The main part (OPCUA client and visualization) is developed using Node-RED⁷.
- Semantic database (GraphDB⁸) were data are stored in triplets.

A semantic representation for each AAS is passed to the integrator so it can use the assets (Figure 7). That way, if an asset changes or updates, a new representation is passed. Then, the integrator communicates with the assets using OPCUA and visualizes their data. That way, plant managers know the current state of the plant and can interact with the different assets. The integrator updates the semantic database on intervals. Moreover, GraphDB offers a SPARQL endpoint so external applications can also query it. Therefore, the different Assets and their properties are also available for external applications.

```
@prefix rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#>
@prefix rdfs: <http://www.w3.org/2000/01/rdf-schema#> .
@prefix owl: <http://www.w3.org/2002/07/owl#> .
@prefix rami: <https://w3id.org/140/rami#> .
@prefix om: <http://www.wurvoc.org/vocabularies/om-1.8/> .
@prefix xsd: <http://www.w3.org/2001/XMLSchema#> .
@prefix il: <http://ikerlan.es/aas#> .

il:AxisXPositionValue a owl:ObjectProperty ;
  rdfs:domain il:SMAxisPosition ;
  rdfs:range om:Measure ;
  rdfs:label "Robotic Arm's X Axis Position"@en ;

il:SMAxisPosition rdf:type owl:Class ;
  rdfs:subClassOf rami:Submodel .

il:RoboticArmAAS rdf:type owl:Class ;
  rdfs:subClassOf rami:AssetAdminShell .

il:RoboticArmAAS1 a il:RoboticArmAAS ;
  rami:hasSubmodel il:SMAxisPosition1 ;
  rami:hasAdminShellId "RA_AAS1"^^xsd:string ;
  rdfs:label "Robotic Arm AAS 1"@en .

il:SMAxisPosition1 a il:SMAxisPosition ;
  il:AxisXPositionValue il:MeasureAxisXPosition1 ;
  il:AxisXSetPositionValue il:AxisXSetPosition1 ;
  il:AxisYPositionValue il:MeasureAxisYPosition1 ;
  il:AxisZPositionValue il:MeasureAxisZPosition1 ;
  rami:subModelID "SM_API"^^xsd:string ;
  rdfs:label "Axis position Submodel 1"@en .

il:MeasureAxisXPosition1 a om:Measure ;
  om:hasUnit om:degree ;
  rdfs:label "X Axis Position in degrees 1"@en ;
  om:hasNumericalValue "161.0"^^xsd:float .
```

⁷ <https://nodered.org/>

⁸ <http://graphdb.ontotext.com/>

Figure 14: Asset semantic representation example using turtle

3.3.2 The practical implications of standardization used in the SOA context.

Since the interoperability of SoS is of high importance to the Eclipse Arrowhead and is perceived as an enabler for seamless integration between components from various manufacturers and on different levels of complexity. This is an incredibly difficult task since the variation of COTS in the field of electronics spans many verticals and horizontals. To achieve the true interoperability, however, the application area should not be the indicator of whether two components or systems can be integrated. Instead, the question “**how they can be integrated?**” should be asked and here standards play an important role.

In the context of SOA subsequent services or application systems exchange information, and that's the point where the integration happens. The lowest layer (e.g. in OSI model) is the physical layer, which is the first step to achieve interoperability - to enable communication between two or more systems, usually using a standardized interface. Without the physical link (wired, wireless or mixed), the data exchange would not be possible. Using standardized interfaces, not only the physical layer is defined, but also higher layers (link, network, transport, for instance) come along as a package, and provide a ready-to-use solution. Here, the first, sometimes overlooked role of the standardization in practical dimension is exhibited. Right now user does not necessarily need to think about how (physically) data would be sent over the network, it is a kind of a plug-and-play feature. Moreover, current technologies integrate various physical layers across SoS, which is an indicator of the desired interoperability.

Another implication of the standardization emerges during the design and integration phases. No matter whether a new system would be deployed or new service/system should be added to an existing one, the need for data exchange forces the designer/designers to determine, how to represent the data in a common way. Fortunately, many application-oriented standards exist and are broadly used to interconnect two independently working components. Ontologies [reference to D4.2] can be a means to have a common dictionary for representing data. Moreover, semantic representation of some data could be a base for implementation of translators between ontologies, which will include also legacy systems in the interoperability landmark.

This could be used also in connection with lifecycle management and modelling for SoS, where the information about membership to a particular engineering phase would give the designer more information about the place in the value chain of a particular system. If there is a catalog of systems ready-to-use, this might speed up the development process and make the integration even easier.

Toolchains reference demo

To show how standardization is useful in practice, the toolchains reference demo developed in WP4 can be used as an example for SoS integration. The architecture of the demo is presented in Figure 15.

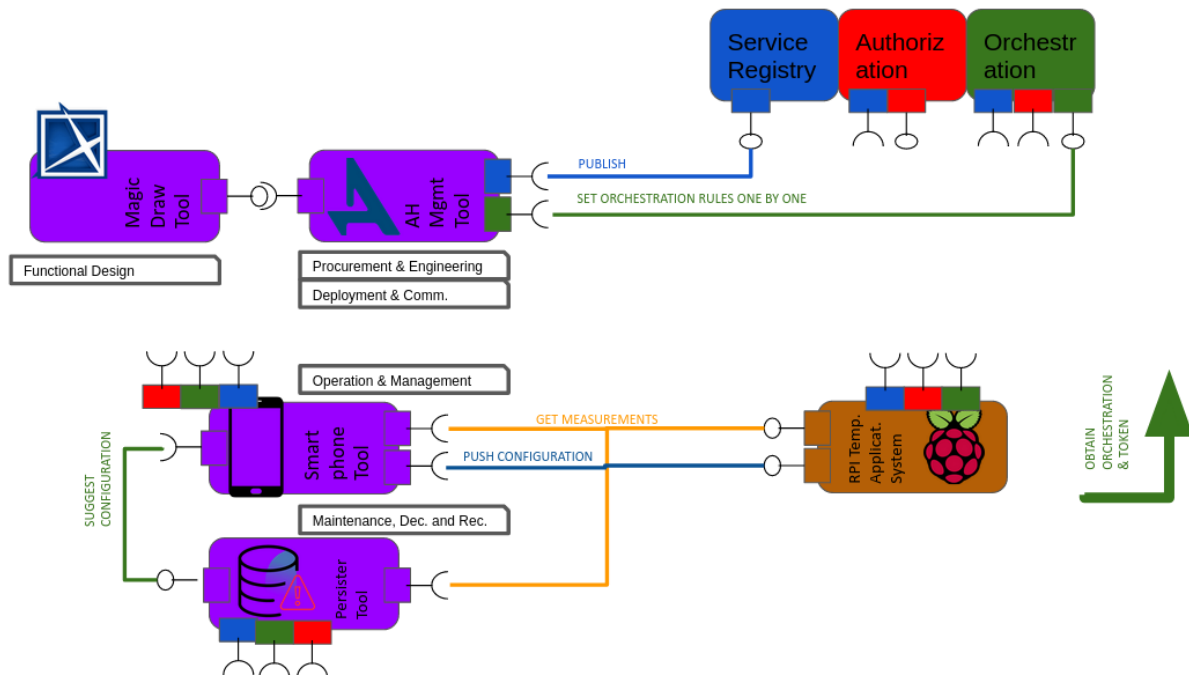


Figure 15: Toolchains reference demo architecture

At first glance the standards are invisible here, but when it comes to integration of the particular services or systems, they play a major role in further data exchange. Most of tools use standardized communication protocols and interfaces, therefore it's quite easy to find a match between tools. Moreover, in Arrowhead's Service Registry, the information about services' interfaces is one of the required metadata when registering a service. In this way, further orchestration, and thus connection between tools, can be matched on the basis of the standards implemented (e.g. JSON, REST API).

Knowing the outgoing or incoming communication interfaces of tools it's possible to design a new one that matches a certain data exchange protocol, which makes further integration seamless. Moreover, when two services are connected with each other, adding a new component between them can be achieved by alignment with the input/output interfaces, and updating the orchestration rules. In this way, building processing pipelines seems to require low engineering effort, and is an advantage of using Eclipse Arrowhead.

Another advantage of using Arrowhead is a set of tools that are included in the framework are supporting core systems. By assigning tools to particular engineering phases, filtering the desired tools by their affiliation to a particular engineering phase provides more information

about their targeted scope of work, and might be useful when integrating with the developing system.

In case two interfaces are incompliant, an appropriate translator or adapter could be included. Thanks to use of standards, a set of most used translators could be predefined and added to the tools catalogue, making the interfacing even easier.

[D42] M. Tatara, F. Montori, et al.: Deliverable D4.2: Arrowhead Tools toolchain design, The Arrowhead Tools Consortium, 2020.

4. Language standards

This chapter is related to the “T10.1 Language standards”. It is focused on contributions to language standards and enhancements of language standards inspired by the needs of Arrowhead, considering a key activity within the process of digitalization of Industrial engineering process to provide full support to complete system lifecycle.

4.1 Language standards in Engineering Industrial Systems

Standards have always been important in industry, but there are many standards for many different purposes. Our purpose in this chapter is to consider the standards for description of our systems with different aims. In general, when it comes to descriptions, often particular description forms are chosen. Even though useful templates are appreciated, documenters frequently introduce their own notation for illustrations as well as for what is intended as a precise textual description. In the Arrowhead Tools project, a major focus is the Eclipse Arrowhead that we would like to promote as a useful and “standard” way of describing and organizing automation situations. Our chapter here is about how Arrowhead relates to other standards of description with different aims.

One obvious aim is to provide a precise, but still intuitive definition of an automation scenario. Arrowhead is dedicated to service-oriented architecture and we would like to find ways to describe such systems with models defined in standard modelling languages. UML and SysML are obvious candidates, and even more so since the interest in SysML has been steadily increasing in the Arrowhead community over the last couple of years boosted also by the profile work initiated by IncQuery Labs. We give more information about this in Section 4.2.1.

At the same time, the Arrowhead community established contact with the emerging new version of SysML and the ongoing standardization through the participation of HIOF. IncQuery Labs and CEA are also Object Management Group (OMG)⁹ members and participating in the SysML v2 Submission Team (SST). We report more on this promising endeavour in Section 4.2.2. With SysML v2 we may also get the chance to define our own notation “the Arrowhead DSL” based on the standard language and its way to define semantics.

SysML v2 also promises novel ways for interoperability of tools. The standard will provide an API for observing and manipulating SysML v2 models which is much more useful and powerful than the old XMI files. The Universidad Carlos III de Madrid has experimented with the pilot implementation of that API and this is described in Section 4.3.

Central to Arrowhead is as just mentioned the need to handle interoperability in heterogeneous scenarios. We cannot require that all the physical and logical machinery that

⁹ <https://www.omg.org/>

we want to include, will adhere directly to the standards we advocate. We need ways to achieve interoperability and exchange of information between very different systems. To that end, the OMG group ManTIS (Manufacturing Technology Interest Group) has started an initiative to facilitate interoperability called MOTIF. This is in its infancy but may become important for Arrowhead in the future. We say something on this in Section 4.4.

Standards is the business of ISO and there are many standards of ISO that are relevant for Industry 4.0, but our focus is on languages and descriptions, and for data modelling, ISO 10303 is of particular interest. Jotne is taking active part in that standardization and applying the standard together with Eclipse Arrowhead and SysML models. We cover ISO 10303 in Section 0.

There are many notations and there are many interpretations of the same notation. To get our arms around such a situation when precise understanding is important for complicated production to succeed, we turn our attention to ontologies as a means to express semantics in ways that can be handled by computers to achieve higher automation and automaticity. We introduce ontologies and their use in Arrowhead in Section 4.6. We also include an overview of ontologies for Industry 4.0 in general found in Section 4.7.

4.2 SysML and UML

4.2.1 The Arrowhead Framework profile for SysML V1

Just as Section 4.2.2 on SysML v2, the present Section is concerned with industry-scale systems modelling, in particular, with the usage of the *Systems Modelling Language* (best known as SysML), a dialect of UML and a standard maintained by the Object Management Group (OMG), for modelling service-oriented System-of-Systems applications and, most importantly, complex Arrowhead instalments.

UML and, in turn, SysML have a built-in language extension mechanism for creating well-defined, specific engineering domains on a conceptual level, using so-called *profiles*. In fact, SysML is realized as a profile of UML. We rely on the profile concept to provide the basic constituents of building service oriented SoS, mostly in an Arrowhead context.

The concepts of the profile, called *stereotypes* in UML/SysML, have been devised independently of any concrete realization, and might, therefore, serve as a conceptual framework for various service-oriented design endeavours. However, in order to align with earlier similar structures proposed in Arrowhead, we adopted some concepts and the naming convention from the *Arrowhead documentation model* (<https://arrowhead.eu/eclipse-arrowhead/this-is-it/documentation-model/>).

The figure below shows the three stereotypes (SD, SD-DD and IDD) used for service modelling in our approach, including their connection to the SysML language. (The figure comes from an actual implementation of the profile in Cameo Systems Modeler.)

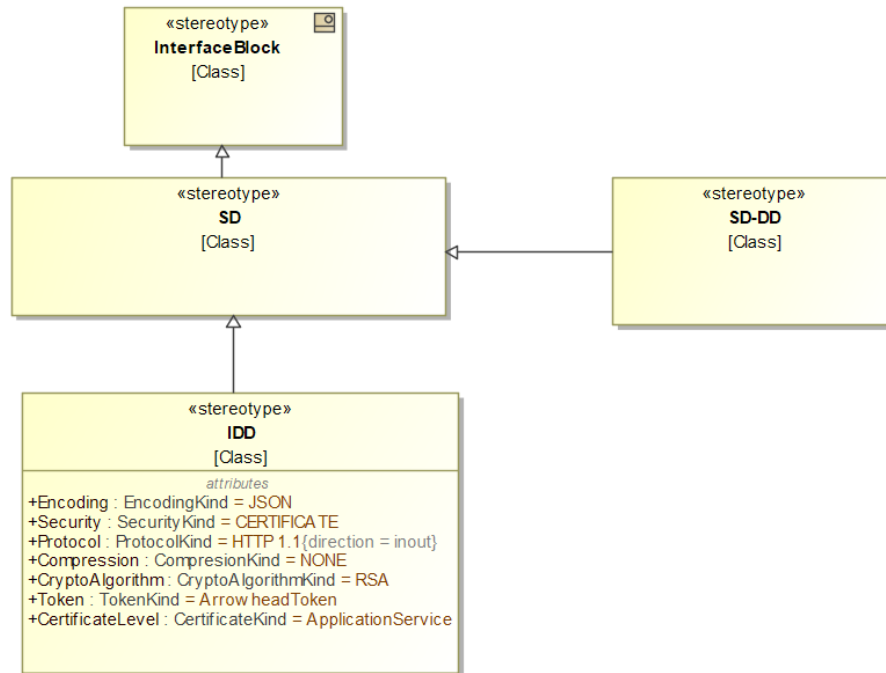


Figure 16: Stereotypes SD, SD-DD and IDD

SD (Service Description) is the abstract unifying concept for services, as generally understood according to the SOA paradigm. In our setting, an SD is a general description of some functionality or data service (e.g., temperature measurements), which can be either provided or consumed by systems.

While an SD represents a black-box perspective on services, its SD-DD (Service Description - Design Description) counterpart captures the realization of a service as it is actually provided (or consumed) by a system.

The actual communication then happens via interfaces, which are represented as IDD (Interface Design Descriptions), i.e., refinements (specializations in UML terminology) of SDs, also providing information on communication details, allowing actual communication to occur between a provider-consumer pair of systems. Here, the applied security, usage of encryption, compression, encoding, etc. are defined, along with the applied payload data model. Instances of IDDs would typically appear as contained in an SD-DD and be interpreted as actual interface endpoints belonging to single operations of an overall service definition.

The next figure (below) shows an overview of the rest of the concepts, namely, stereotypes concerned with systems and compound entities based on them, local clouds as well as systems of local clouds.

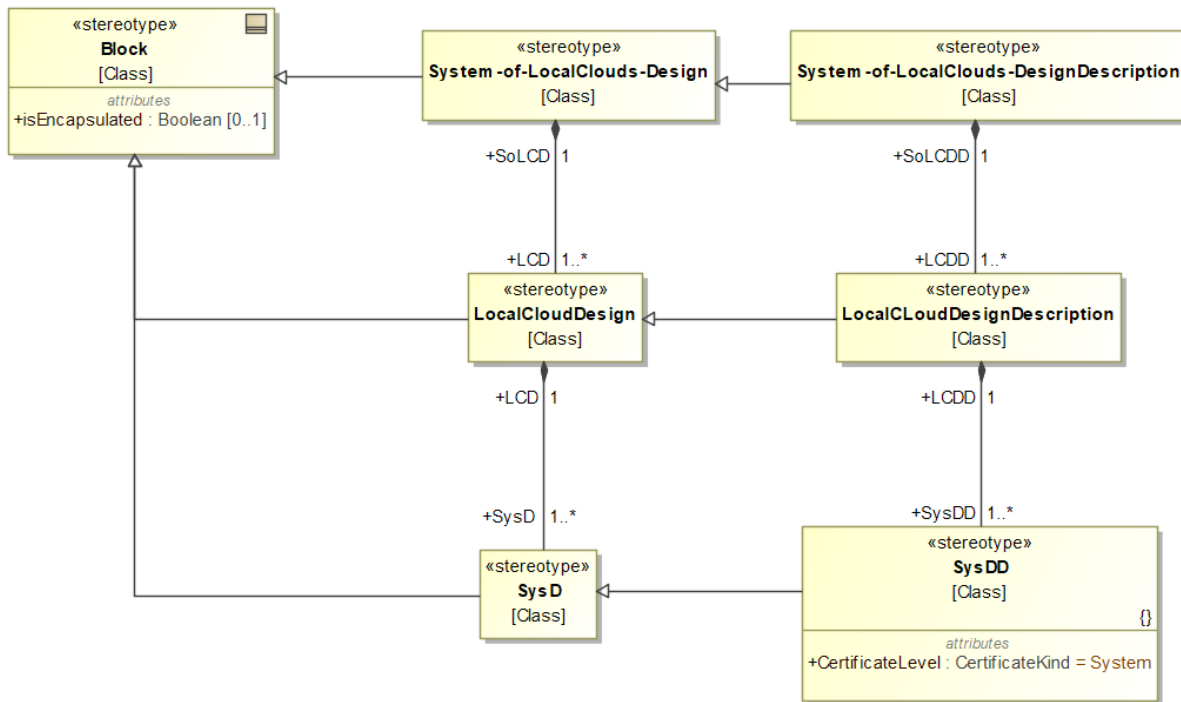


Figure 17: System and Cloud Stereotypes

The SysD (System Design) stereotype represents a black-box view of systems, i.e., it can be considered a generic template for (the software behaviour of) a given system type in the SoS that we want to model. The SysDD (System Design Description) stereotype represents a white-box view of systems, resulting in higher precision regarding the details of the behaviour. Note, however, that these system stereotypes can be used hierarchically, i.e., there are no strict constraints on how much information has to be provided in a single SysDD, and the amount of information can be refined in further specialization blocks.

A very important, even central, usage of SysD and SysDD blocks is to hold ports (which are, again, basic SysML elements), i.e., connections with the outer world. However, in SysML, the exact interpretation of a port is left to the modeler. The interpretation in our profile is that a so-called full port is typed over an SD, representing an overall service provided or consumed by that system, while the individual interfaces of that service are modelled by IDD-typed proxy ports contained in that full port.

An update of it can be seen in Figure 18. An Arrowhead adaptor and a sequence diagram of how OPC-UA services are made available to the wider Arrowhead interoperability. It includes engineering sequence proposal from manuscript [40] on Smart City engineering considering SoS engineering based on SysML. The integration is made with the Engineering process of WP4 and the basic Eclipse Arrowhead architecture levels.



Figure 18: Arrowhead adaptor and a sequence diagram of how OPC-UA services are made available to the wider Arrowhead interoperability

The SysD (System Design) stereotype represents a black-box view of systems, i.e., it can be considered a generic template for (the software behaviour of) a given system type in the SoS that we want to model. The SysDD (System Design Description) stereotype represents a white-box view of systems, resulting in higher precision regarding the details of the behaviour. Note, however, that these system stereotypes can be used hierarchically, i.e., there are no strict constraints on how much information has to be provided in a single SysDD, and the amount of information can be refined in further specialization blocks.

A very important, even central, usage of SysD and SysDD blocks is to hold ports (which are, again, basic SysML elements), i.e., connections with the outer world. However, in SysML, the exact interpretation of a port is left to the modeler. The interpretation in our profile is that a so-called full port is typed over an SD, representing an overall service provided or consumed by that system, while the individual interfaces of that service are modelled by IDD-typed proxy ports contained in that full port.

A local cloud is an SoS instance, where the partaking systems have a certain level of trust towards each other, enabling a more direct orchestration mechanism between them, while at the same time, the local cloud also implies a logical grouping of its constituents. As expected, the LocalCloudDesign stereotype corresponds to other Design stereotypes and serves as their logical container, while the LocalCloudDesignDescription is a more detailed entity with a white-box view on design details, as represented by other Design Description stereotypes within. Analogously, the System of Local Cloud (SoLC) stereotypes also build a pair, where an SoLC SoS instance, having a compound, more global nature.

Besides providing a conceptual framework for reasoning about service-oriented SoS design within and beyond Arrowhead, the profile serves a foundation for important operational and interoperational considerations in the Arrowhead Tools project.

The reference implementation of the profile, realized in Cameo Systems Modeler, has been extended by IncQuery Labs with a set of end-user tools to (i) transform models into real-life Arrowhead components via the Arrowhead Management Tool, (ii) model their deployments to device digital twins via Eclipse Vorto, and (iii) to provide an abstract description of Arrowhead toolchains within the same tooling. This toolkit is available at <https://github.com/IncQueryLabs/arrowhead-tools-magicdraw-plugins>.

In order to foster a wider reach for the profile, an adaptation of the reference implementation is now available for the users of Eclipse Papyrus, performed by CEA. An important further feature provided in this line of work is an *OpenAPI-based code generator* to create Arrowhead client code in various languages, from an adequately extended IDD concept.

A central usage of the profile is an *Arrowhead core library*, created by Jerker Delsing, featuring a full-fledged model of all the Arrowhead core systems based on the profile,

implemented in Cameo Systems Modeler. The profile and the library will be available in the GitHub repository of Eclipse Arrowhead.

4.2.2 The Arrowhead Validation use case on Service-Oriented Architecture in SysMLv2

SysML v2 is a remake of SysML which may also prove to be a remake of system modelling languages. When the interest for applying modelling languages for Arrowhead increased, it became probable that participating in the OMG standardization of the new SysML standard could be advantageous. The timing seemed perfect since SysML v2 is scheduled to finalize approximately at the same time as Arrowhead Tools.

Object Management Group (OMG) has been the main standardizing body for modelling languages since the late 1990-ies when the Unified Modelling Language (UML) appeared as a combination of the languages of the “three amigos”, Jim Rumbaugh, Grady Booch and Ivar Jacobson. In early 2000 an effort to create a new and more versatile version of UML appeared and became UML 2.0 and at the same time the need for a system modelling language evolved into the language SysML driven by Sanford Friedenthal and the first SysML technology was available in 2005.

The SysML v2 efforts have been going on for several years already and a considerable collection of requirements has been collected and two Requests For Proposals were created and the formal OMG process started. The language RFP was issued in December 2017 and the RFP for API and services was issued half a year later. Please note that the services requested in that RFP are not services intended for specifying service-oriented architecture in SysML itself. That RFP is for letting tools get access to SysML v2 models in a more integrated and efficient way than what has been the case when using XMI as the exchange format.

Now there is only one consortium working on the development of SysML v2, called SST (SysML Submission Team) managed by Sanford Friedenthal and Ed Seidewitz. The consortium now has around 75 OMG members as partners, and more than 100 persons participate. The Arrowhead Tools project also have members in that consortium. IncQuery contributes as providers of pilot implementation pieces (Track 6), and CEA is a general partner inspired by their role as tool vendor of Papyrus. HIOF (Østfold University College) contributes to Track 2 on requirements and Track 4 on the metamodel. There are in particular three issues where HIOF has brought Arrowhead to the attention of the SST and the SysML v2 efforts. Øystein Haugen of HIOF has been working in the OMG since 2000 and participated in a series of standards often related to UML 2.x.

Having spent some time getting to know the work behind the SysML v2, we became aware that there were very few requirements to SysML v2 from the domain of service-oriented architectures (SoA) which is what Arrowhead Framework applies as its paradigm.

Therefore, we suggested to introduce a validation use-case in the SysML v2 consortium SST in the domain of SoA. The use-case has the Norwegian use-case of Productive4.0 as starting point, but reaps the benefit of the SysML v1 profile work (Section 4.2.1) to make a SysML v2 library for Arrowhead Framework.

Our three ongoing discussions in the SST are these:

- The general SoA architecture and service functionality based on nested ports. Synchronous communication through remote method calls as well as asynchronous communication through signalling.
- Language extension such that we can express our models in Arrowhead terms and concepts and still rely on SysML v2 constructs and semantics.
- Sequence diagrams to express service interaction in a system and a system of systems.

The general SoA architecture is directly inspired from the SysML v1 profile and in SysML v2 the profile information is expressed as a library. The SysML v2 is expressed as a textual modelling language during the development of the language supported by an evolving pilot implementation which is open source¹⁰. The pilot implementation comes both as a plugin of Eclipse, and also as an app on Jupyter Lab.

¹⁰ <https://github.com/Systems-Modeling/SysML-v2-Release>

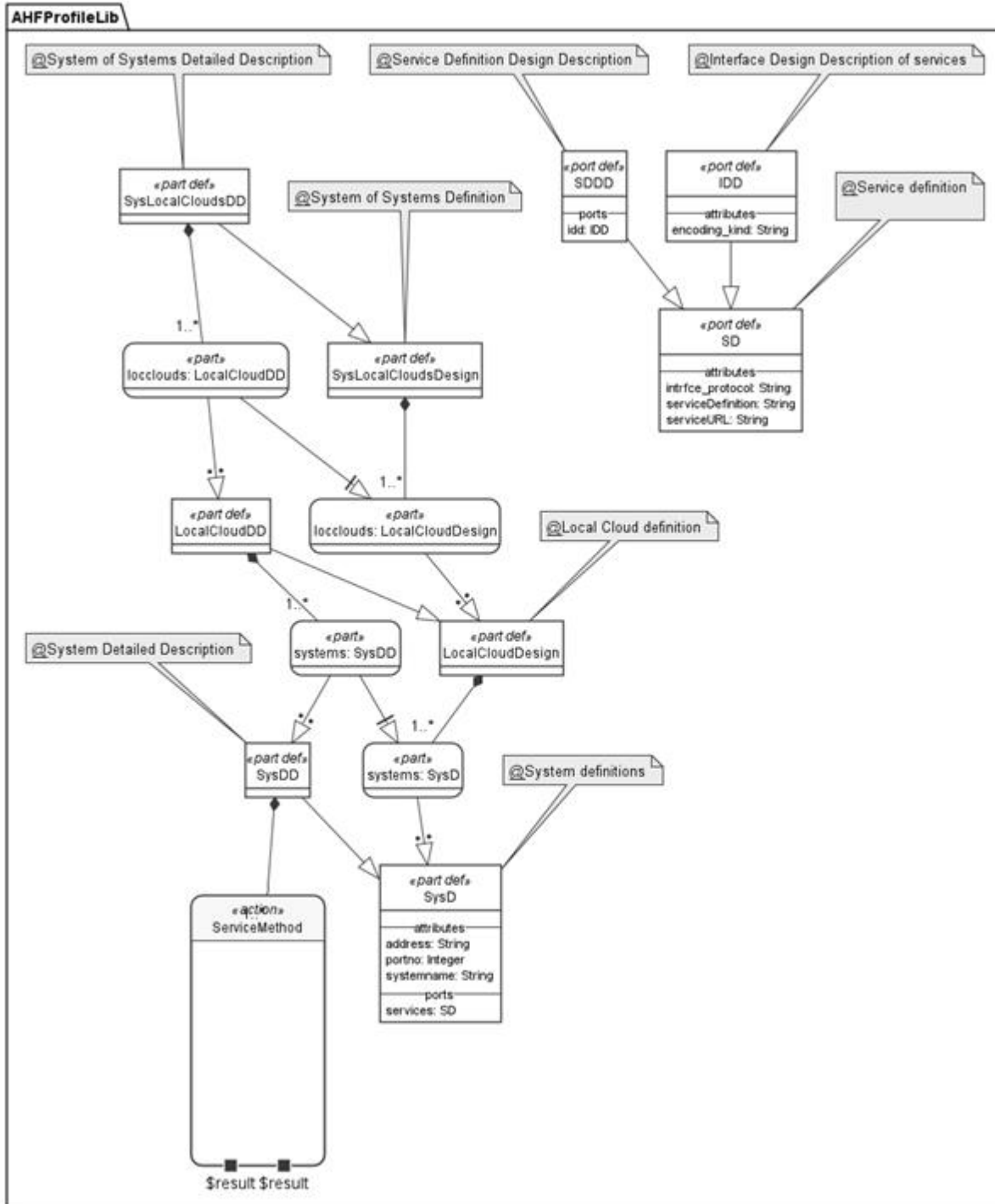


Figure 19: Arrowhead library in SysML v2 corresponding to Arrowhead profile

The textual notation can be directly visualized through PlantUML and this is what has been shown in Figure 19 . By applying the API which is also a pilot implementation, there is in addition another more advanced rendering provided by the partner Tom Sawyer.

The Arrowhead validation use-case (Figure 19) was first applied to show the need for expressing REST-oriented communication as well as asynchronous communication like MQTT. Together with IBM we have been discussing the details of how this should appear in the language definition to make service-oriented architecture an attractive paradigm supported by SysML v2.

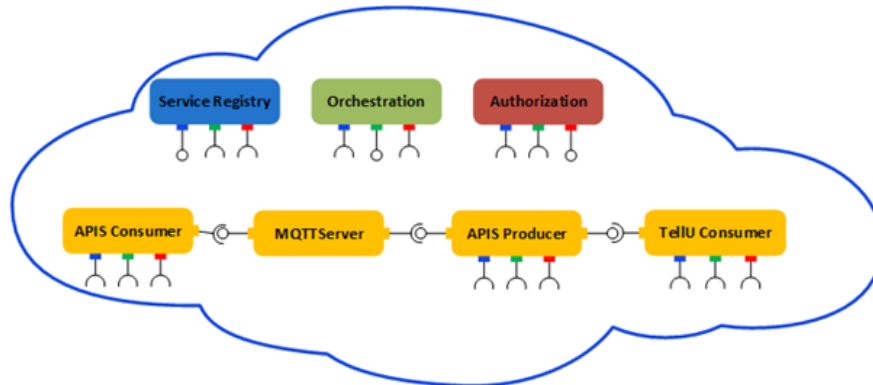


Figure 20: The Norwegian use-case from Productive4.0 in Arrowhead notation.

The way we have modelled the Arrowhead use-case in SysML v2 turns out to be very suitable for creating a domain-specific languages extension based on the same principles as SysML v2 itself built from KerML. In Figure 21 we see an excerpt of our mock-up Arrowhead DSL description of the use-case.

```
#cloud def AHFNorway_SoSD { // Arrowhead local cloud is a part def specialized from ArrowheadCore
#system def TellUConsumer { // Arrowhead system defs are part defs specialized from SysDD
#service serviceDiscovery:~ServiceDiscovery // service usages subsets services of AHE
#service apisp:APIS_REST
{
  /** The body here is to get the contexts and types right, may be changed later */
  require action :>> giveItems;
  provide action :>> getAllItems;
}
```

Figure 21: Mock-up prototyping of an Arrowhead DSL description

We are now working on including sequence diagrams into SysML v2. Since Arrowhead will apply nested ports to describe service functionality, we should have sequence diagrams with decomposition of lifelines such as shown in Figure 22. This has not yet been included in the current version, but we are working to integrate sequence diagrams with actions and states better than what is the case in UML 2. One of the main goals of SysML v2 is to have a uniform semantic base for behaviour descriptions.

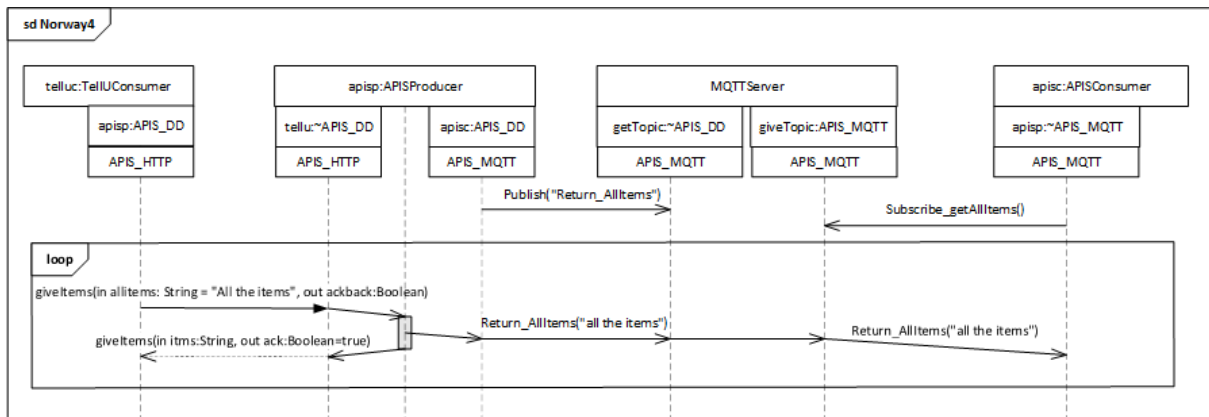


Figure 22: Sequence diagram with inline nested ports

The SST process intends to reach a revised submission for the OMG meeting in September this year, and that will hopefully send SysML v2 into finalization which will take yet another year.

4.3 The Arrowhead Tools experiment on SysML v2 API.

As it has been presented in WP2, the use case 3 (UC-3) aims at providing means for improving the reuse of physical models covering the abstraction, selection, representation, and customization of system artifacts for the whole development lifecycle. The reuse of any system artifact goes beyond the mere discovery of a potential reuse and it must focus on evaluating what and how a system artifact can be reused (requirements, analytical models, descriptive models, test cases, etc.). To do so, quality also plays a role since it is assumed that high-quality system artifacts may help to improve the reusability factor of a system artifact. Furthermore, in this use case, there is another major objective focusing on the improvement of traceability to be able to automatically keep traces from [41] the very early stage of development to the final release of a complex product. In this context, different technical engineering processes and engineering methods (supported by different techniques and tools) build the next toolchain.

Engineering process	Engineering methods	Techniques	Tools
System Requirements Definition	Requirements Engineering	Natural language requirements	IBM Doors, Requirements Authoring Tool
Architecture Definition	Logical modelling	Diagramming with SysML/UML	IBM Rhapsody
Design definition	Physical modelling	Diagramming	Altium designer

Engineering process	Engineering methods	Techniques	Tools
Implementation	Simulation	Programming, simulation configuration	Altium designer, native code
Verification & Validation (Measurement process)	Quality management	Quality metrics	Verification Studio
Information Management	Knowledge engineering	Ontologies	Knowledge Manager (KM)
Information Management	Knowledge management	Traceability discovery	KM

Table 2: Summary of engineering processes, methods, techniques and tools for UC3.

In general, there are three tools providers: IBM, The Reuse Company and Altium. Most of them provide standardized ways of accessing (files and services) and consuming work products data and operations. However, the interpretation of standards (such as SysML or ReqIf) may differ from one tool to another and, in most of cases, the tools also manage more relevant information that is not exposed being critical for processes such as traceability or quality management.

4.3.1 Background on logical modelling tools

In terms of the engineering process and system development lifecycle, traditional linear approaches, e.g. the Waterfall or *Vee* models, or iterative models like Concurrent Engineering have been re-designed to support the digitalization of the engineering process and accommodate them to the new needs of the Industry 4.0 [42]. The Digital Thread [43] by the Boeing Company has introduced the concept of a diamond to provide a digital version of the well-known *Vee* model linking the physical and virtual worlds. This new development lifecycle makes intensive use of digital assets in form of models as unit of exchange and simulation techniques to enable the notion of digital twin through automation and collaborative engineering [44].

Other attempts to improve the system engineering process and development lifecycle can be found in the Future of Systems Engineering Roadmap [45] promoted by the SERC (Systems Engineering Research Center). This document establishes five major goals to strategically change the engineering process: 1) model use for decision making, 2) authoritative source of truth, 3) technology innovation, 4) collaborative environment and 5) digital engineering (DE) workforce and culture.

More specifically, goal 2 “Authoritative Source of Truth” includes data integration/interoperability framework, DE (Digitalization of Engineering) design process, semantic data links and digital twin innovation as a path to reach an augmented engineering process. Goal 3 also includes semantic web data exchange and inter-enterprise data integration as key-enablers for improving the engineering process while goals 4 and 5 also stand out DE Change Management, Authoritative Data identification and Process & Methods as other cornerstone elements of the augmented engineering process.

The definition of a strategy for DE strongly relies on a technological support implemented through different tools that must be seamless orchestrated (toolchain) to provide new collaboration environments. In this sense, interoperability and standardization play a key role to boost the current engineering practice connecting silos of organizations, people, data, information and knowledge easing the design, creation and evolution of complex systems in a timely, safe and cost-effective manner.

In summary, these attempts to improve the engineering practice emerge to improve the:

- 1) digitalization of engineering work products (digital twin),
- 2) creation of collaborative engineering environments and
- 3) automation and communication at different levels of abstraction: people (e.g. engineering teams), organizations (e.g. manufacturers-suppliers), engineering stages/activities and methods (e.g. requirements engineering and verification/validation) and tools (e.g. requirements management system and logical modelling tools).

Additionally, standardization remains as a key-enabler to address in both development and operation stages. In fact, one of the means to reach a proper level of automation and communication within the development lifecycle relies on providing different levels of interoperability enabling the communication and exchange of data, information and knowledge between people, organizations and tools.

In the frame of complex systems development, the use of architectural frameworks, standardized languages, common data models and communication protocols is a common practice to enable systems interoperability in both sides: development and operation. In this context, last times have seen the emergence of Model-based Systems Engineering (MBSE) as a complete methodology to address the challenge of unifying the techniques, methods and tools. This means a “formalized application of modelling” to support the left-hand side in the *Vee* lifecycle model implying that any process, task or activity will generate different system artifacts but all of them represented as a model.

The MBSE approach is considered a cornerstone for the improvement of the current practice in the Systems Engineering discipline since it is expected to cover multiple domains [46], to provide better results in terms of quality and productivity, lower risks and, in general, to support the concept of continuous and collaborative engineering. However, the MBSE

approach considers that everything can be a model, e.g. SysML or physical model, and this assumption is not always true in the development of a complex system.

Currently, interoperability initiatives for complex systems development such as the family of standards ISO 10303-STEP or OASIS OSLC (Open Services for Lifecycle Collaboration), are trying to boost interoperability through a common data exchange model based on unifying data models and communication protocols, e.g.[47] . Both define a collaborative engineering ecosystem through the definition of data shapes or schemes that serve us as a contract to get access to information resources. The Representational State Transfer (REST) software architecture style is used in both to manage information resources that are publicly represented and exchanged in different formats such as JSON or XML. Last draft of SysMLV2 (Systems Modelling Application Programming Interface and Services specifications) [48] is also following a similar approach defining a REST API to consume SysML models.

4.3.2 SysMLV2 introduction

The System Modelling language (version 2, SysMLV2) has emerged as a new modelling language oriented to cover the necessities of the Systems Engineering discipline. More specifically, it is one of cornerstones languages for descriptive models creating a real digital twin of complex systems in combination with other languages for analytical modelling (e.g. Modelica or Simulink).

Initially, SysML was designed as a profile of UML to reuse the existing tools and take advantage of the experience of logical modelling in software systems. Since logical modelling becomes a key activity in designing complex systems through model-based methodologies, the new version brings us most of the required capabilities to support the technical engineering process of “specification, analysis, design and verification and validation including software, hardware, information, processes, personnel and facilities”.

The key features of the SysMLV2 relies on a new metamodel that is not constrained by UML and grounded in a formal semantics, robust visualizations based on views and viewpoints and a standardized API to access the model elements. According to the specification, it is also possible establish some relevant differences in regard to the previous version: additional functionalities (e.g. variants), integration of concepts between structural and behavioral elements, ease to use and clarification of concepts (e.g. individuals vs instances).

In the context of the AHtools project and the UC-3, the main feature of the SysMLV2 (apart from those regarding expressivity and design of the language) is the possibility of using an API to get access to model elements. Based on previous experiences consuming logical models from different tools (see the next Table) like IBM Rhapsody, Magic Draw, Capella or Papyrus, we have found issues due to the different serializations of the same model in XMI. In terms of writing, the same issue remains since it is complicated to provide a real exchange mechanism between tools. Moreover, some of these tools also offer the visual representation of models but it is not completely standardized. The first approach to

overcome these issues was the creation of native connectors (usually in the form of tool plugins) that could internally connect to the structures managed by the tools and get all information from the models (including the possibility of invoking operations). However, this approach comes with a high cost due to the need of learning a new API and an object model for each new tool.

Furthermore, in terms of maintenance, new updates in the tools also require a new plugin version which implies again a high cost to keep different versions up and running. Considering the high costs of development and maintenance and the lack of interoperability (communication, syntax and semantic) in logical models, the possibility of consuming a standardized SysML API opens new possibilities up, delegating these efforts to the tool providers and enabling consumers to easily connect to the logical modelling tools.

		Rhapsody 8.1.4	MagicDraw/Cameo	Papyrus
Native information storage file		The tool does not offer the possibility of storing the native file in XML format.	The tool offers the possibility of storing the native file in XML format (see https://docs.nomagic.com/display/MD182/Saving+projects).	The native papyrus file (.uml) is developed using XML and can be imported from other tools.
Capability to export the model to XML and UML versions		UML2.1, UML2.2 and UML2.3 (see https://www.ibm.com/support/knowledgecenter/SSB2MU_8.2.0/com.ibm.rhp.oem.pdf/doc/pdf/sodiux/XML_Toolkit_User_Guide.pdf)	UML XMI 2.4 file, EMF Ecore file, MOF XMI file and Eclipse UML2 (v1.x, v2.x, v3.x, v4.x, v5.x) XMI file (see https://docs.nomagic.com/display/MD190/Exporting+UML+models)	It does not allow us any type of export, one of the native Papyrus files contains model information using an XML-based standard
<p>46 / 5000</p> <p>Some differences between XMI representations</p> <p>The XML tags are represented differently in some of the tools, below are examples for Association, Dependencies, Abstraction objects.</p>	Association / Composition / Aggregation	<pre><packageElement xmi:type="uml:Association" xmi:id="19_0_3_6bb0192_1608131004406_123950_1953" memberEnd="GUID+ac3e0225-0d17-45d0-bb62-8fcf2e836b10 GUID+ba2884-0049-4b2a-b7a9-d7c692dc79a1" navigableOwnedEnd="GUID+ac3e0225-0d17-45d0-bb62-8fcf2e836b10" > <ownedEnd xmi:type="uml:Property" xmi:id="GUID+ac3e0225-0d17-45d0-bb62-8fcf2e836b10" name="ItsFireControl" visibility="protected" type="GUID+af6c836e-25a4-4692-ba6f-b80372538875" aggregation="composite" owningAssociation="MTHQ4NSee_O59pkcneEQ" association="MTHQ4NSee_O59pkcneEQ"/> </packageElement></pre>	<pre><packageElement xmi:type="uml:Association" xmi:id="19_0_3_6bb0192_1608131004406_123950_1953" memberEnd xmi:idref="19_0_3_6bb0192_1608131013339_6378_3780"/> <memberEnd xmi:idref="19_0_3_6bb0192_1608131013339_579127_3781"/> <navigableOwnedEnd xmi:idref="19_0_3_6bb0192_1608131013339_579127_3781"/> <navigableOwnedEnd xmi:idref="19_0_3_6bb0192_1608131013339_6378_3780"/> <ownedEnd xmi:type="uml:Property" xmi:id="19_0_3_6bb0192_1608131013339_6378_3780" visibility="public" type="19_0_3_6bb0192_1608131004364_23403_3884" association="19_0_3_6bb0192_1608131004406_123950_1953"/> <ownedEnd xmi:type="uml:Property" xmi:id="19_0_3_6bb0192_1608131013339_579127_3781" visibility="public" type="19_0_3_6bb0192_1608131004403_834207_1951" association="19_0_3_6bb0192_1608131004406_123950_1953"/> </packageElement></pre>	<pre><packageElement xmi:type="uml:Association" xmi:id="_3EeqBHCkEeu_66b57EN3Lw" memberEnd="_3EtsUHCkEeu_66b57EN3Lw_3Eu7YnCKEeu_66b57EN3Lw"/> <eAnnotations xmi:type="ecore:EAnnotation" xmi:id="_3EseMHCKEeu_66b57EN3Lw" source="org.eclipse.papyrus"> <details xmi:type="ecore:EStringToEntry" xmi:id="_3EseMXCKEeu_66b57EN3Lw" key="nature" value="UML_Nature"/> </eAnnotations> <ownedEnd xmi:type="uml:Property" xmi:id="_3Eu7YnCKEeu_66b57EN3Lw" name="class13" type="_qZCqBHCkEeu_66b57EN3Lw" aggregation="composite" association="_3EqBHCkEeu_66b57EN3Lw"/> </packageElement></pre>
	Dependency	<pre><packageElement xmi:type="uml:Dependency" xmi:id="GUID+42bbc76c-7212-4734-ac6b-a132dfe913f" name="dBTS" supplier="GUID+2d5c5c08-ab4e-41f9-819f-ef0c4555bcd6" client="GUID+d9f5a654-6204-4a52-8766-521425ae6cc2"/></pre>	<pre><packageElement xmi:type="uml:Dependency" xmi:id="19_0_3_6bb0192_1608131004408_132127_1958" client xmi:idref="19_0_3_6bb0192_1608131004403_834207_1951"/> <supplier xmi:idref="19_0_3_6bb0192_1608131001355_431772_1064"/> </packageElement></pre>	<pre><packageElement xmi:type="uml:Abstraction" xmi:id="_HXymUHCyEeu_66b57EN3Lw" name="Trace19" client="_GvkiSHCYEeu_66b57EN3Lw" supplier="_GwaB0HCyEeu_66b57EN3Lw"/></pre>
	Abstraction	<pre><packageElement xmi:type="uml:Abstraction" xmi:id="GUID+e3bbd2b7-e8b3-45d3-9678-c92cf27191a" name="Req_3_2" supplier="GUID+1c4166a-797c-4a73-a79f-3d5ce524aad0" client="GUID+23b9ae0d-684b-42da-9f40-e283f09d18fe"/></pre>	<pre><packageElement xmi:type="uml:Abstraction" xmi:id="19_0_3_6bb0192_1608131006585_68333_2273" client xmi:idref="19_0_3_6bb0192_1608131006542_109437_2209"/> <supplier xmi:idref="19_0_3_6bb0192_1608131004387_357665_1922"/> </packageElement></pre>	<pre><packageElement xmi:type="uml:Abstraction" xmi:id="1DjUHCwEeu_66b57EN3Lw" clients="1eeuLHCkEeu_66b57EN3Lw" supplier="_qZCqBHCkEeu_66b57EN3Lw"/></pre>
Are diagrams included?		No	Yes	No
Other items included		Rhapsody profiles and annotations	MagicDraw/ Cameo profiles and annotations	Papyrus profiles and annotations

For more information about the interoperability of tools, see the work referenced at[49].

4.3.3 SysMLV2 API model overview

The SysMLV2 API [48] model assumes a 3-layer description. The first layer, the Platform-Independent Model (PIM) provides a service specification consistent with the KerML and SyML. It represents a logical API model providing a service specification completely independent from the specific implementations. The second layer, Platform-Specific Models (PSMs) provides the binding between the PIM and the specific technological implementations such as REST/HTTP, SOAP, Java or .NET. Two main PSMs are already available: 1) REST/HTTP, a binding based on the OpenAPI specification and 2) OSLC PSM, a binding to accommodate the descriptions following the OSLC-based standards.

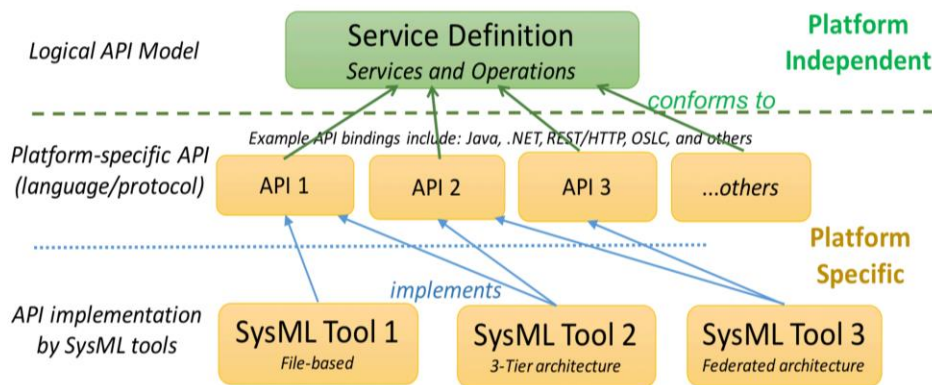


Figure 1. API and Services Architecture

Figure 23: SysMLV2 API models (source: SysMLV2 specification [48]).

The logical API model mainly contains the following entities: project, element, relationships, commit and query. This model is quite similar to others that can be found in tools like Atlassian Bitbucket, MMS (Model Management System) but including configuration management capabilities.

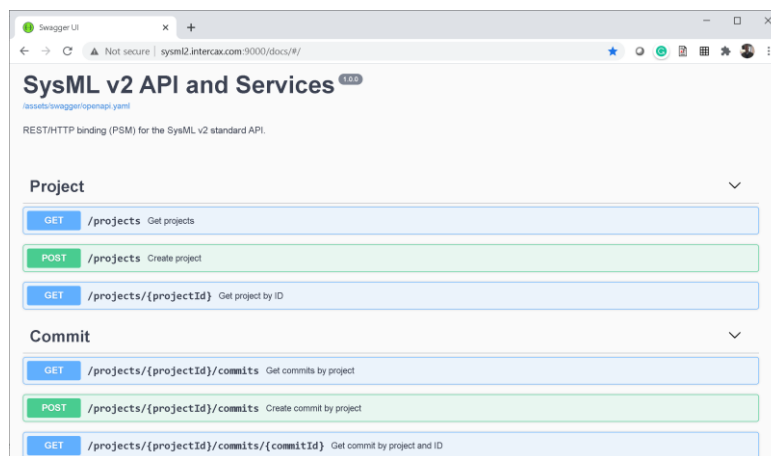


Figure 24: SysMLV2 API PSM based on the OpenAPI specification (source: endpoint provided by Intercax).

Going through the different elements, we highlight here the main operations and object models (extracted from the official documentation):

1. Project operations.

Name	Documentation
update_root_elements	Sets the given elements as the root elements of the given project. Precondition: Project and elements must be specified and valid.
create_project	Creates a new project with the given name and description. Precondition: Name and description must be provided and valid.
get_project_by_id	Get project with the given identifier. Precondition: Identifier must be specified and valid.
get_projects	Gets all projects.
get_root_elements	Gets all the root elements for the given project. Precondition: Project must be specified and valid.

Figure 25: Project API operations (source: SysMLV2 official documentation [48]).

2. Element versioning operations and object model.

Name	Documentation
get_commit_by_id	Gets a specific commit in a project by the commitID. Precondition: Project and commitID must be specified and valid.
get_commits	Gets all the commits in a given project. Precondition: Project must be specified and valid.
create_commit	Creates a new commit given the change set (collection of all elements that have updated). Precondition: One or more elements with updates must be specified and valid.
get_latest_commit	Gets the latest commit in a given project. Precondition: Project must be specified and valid.
get_previous_commits	Gets all the previous commits in order (latest to oldest) given a commit. Precondition: Commit must be specified and valid.
get_versions	Gets all versions of the given element. Precondition: Element Identity must be provided and valid.

Figure 26: Configuration management (versioning) API operations (source: SysMLV2 official documentation [48]).

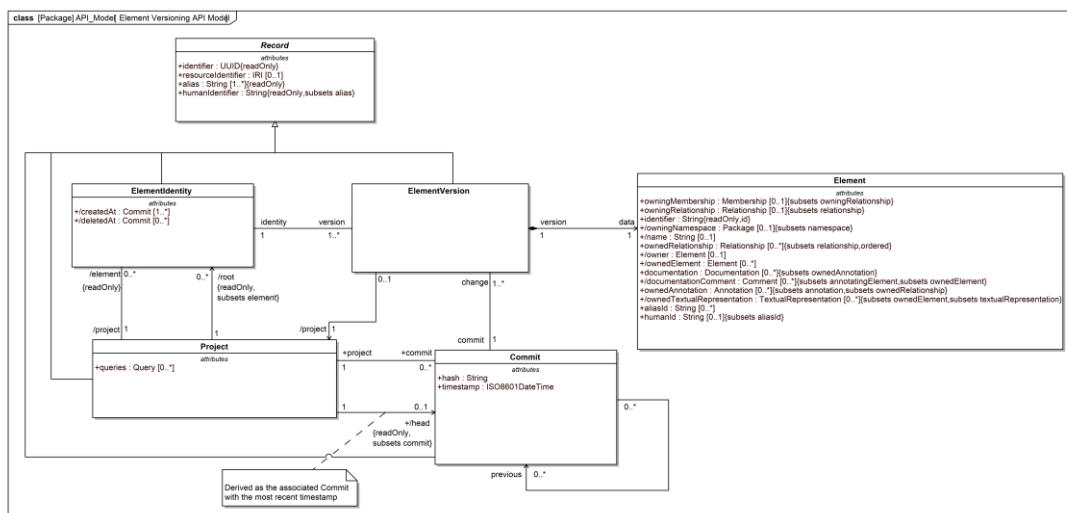


Figure 27: Element and configuration management (versioning) API model (source: SysMLV2 official documentation [48]).

3. Element and relationship model.

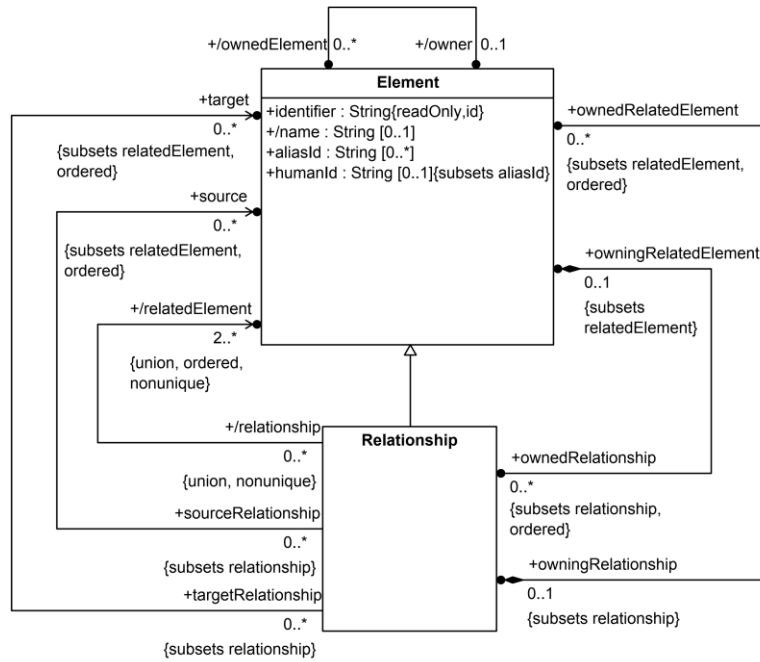


Figure 28: Element and relationship API model (source: SysMLV2 official documentation [48]).

Name	Documentation
get_all_elements	Get all the elements in a given project at a given commit. Precondition: Project and Commit must be specified and valid.
get_element_by_id	Gets the element with the given identifier. Precondition: Identifier (UUID) for the element must be provided.

Name	Documentation
get_element_by_type	Gets elements with the given type. If project is specified, gets elements with the given type in that project. Precondition: Element type must be provided and valid. Project must be valid.
get_relationships_by_source_target_element	Gets relationships by source element. If project is specified, gets relationships with the given source element in that project. Precondition: Source element must be specified and valid. Project must be valid.

Figure 30: Element and relationship API operations (source: SysMLV2 official documentation [48]).

4. Query model.

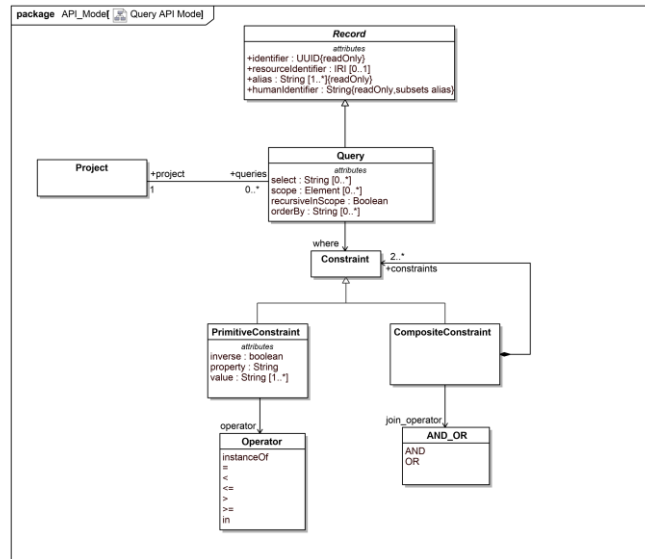


Figure 31: Query API model (source: SysMLV2 official documentation [48]).

Name	Documentation
execute_query	Executes the given query and returns a collection of Elements. Precondition: Query must be provided and valid.
create_query	Creates a query in a project, given the query inputs. Precondition: Project and where constraint must be specified and valid.

Figure 32: Query API operations (source: SysMLV2 official documentation [48]).

4.3.4 Consuming the SysMLV2 API: example of output

The next table shows some outputs of the main operations offered by a SysMLV2 endpoint linked through a project ID (entry point to consume the different resources and operations).

API Element	Output
Project API	<pre>[{ 'id': '9fb5e180-7798-498f-a32c-feeec7bb9ab', 'name': '3a-Function-based Behavior-1 Fri Oct 30 01:13:33 EDT 2020', 'type': 'Project' }]</pre>
Commit API	<pre>[{ 'change': None, 'containing_project': {'id': '9fb5e180-7798-498f-a32c-feeec7bb9ab'}, 'id': '3c5484db-b179-46f4-8e58-ed56bfa8bb03', 'previous_commit': None, 'type': 'Commit' }]</pre>
Element API	<pre>Elements 479 {'ItemFlowEnd', 'Comment', 'AcceptActionUsage', 'FeatureTyping', 'Subsetting', 'ActionUsage', 'Step', 'Import', 'ParameterMembership', 'EndFeatureMembership', 'ItemFeature', 'Superclassing', 'BindingConnector', 'ItemFlowFeature', 'Package', 'Feature', 'FeatureMembership', 'Succession', 'ReferenceUsage', 'Membership',</pre>

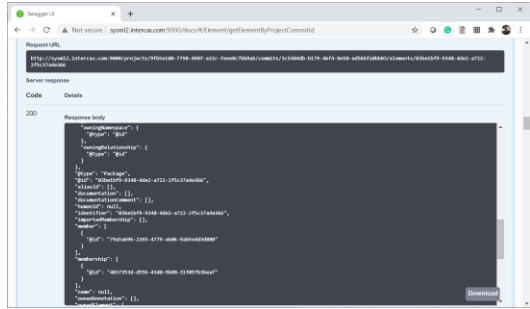
	<p>'ActionDefinition', 'AttributeDefinition', 'Annotation', 'MergeNode', 'DataType', 'Redefinition', 'ItemFlow')</p>
<p>Element API</p>	 <p>http://sysml2.intercax.com:9000/projects/9fb5e180-7798-498f-a32c-f6ee8c7bb9ab/commits/3c5484db-b179-46f4-8e58-ed56bfa8bb03/elements/03be1bf9-9348-4de2-a722-2f5c37a4e366</p>

Table 3: Some outputs of the main operations offered by a SysMLV2

4.3.5 Evaluation: status and plan

The initial work with the SysMLV2 API has been focused on knowing the object models and operations. To do so, a Python client has been developed to consume the information offered by the Intercax service. Next steps must cover:

- Exposing the logical models of the UC-3 through a SysMLV2 API
- Exposing the SysMLV2 API through the Arrowhead Framework
- Consumption of the UC-3 models through .NET (a client of the SysMLV2 API in .NET)
- Consumption of the UC-3 models through .NET (a client of the SysMLV2 API in .NET through the Arrowhead framework)
- Report the results to the SysMLV2 standardization effort

Assuming that official implementation of the SysMLV2 API will not be available in the tools (e.g. IBM Rhapsody), we expect to be able to start a service indicating the logical model to be offered. At least, the possibility of reading models should be implemented.

On the other hand, this first attempt to learn the SysMLV2 API also allows us to evaluate the approach from different perspectives:

- Conceptual level. SysMLV2 is a language that has been designed for Systems Engineering covering most of the needs in this discipline for logical modelling and linking to analytical models. In the case of TRC (The Reuse Company), the SysML element and relationship model seems quite similar to their own object model (the System Representation Language) so, the mapping between SysML and System Representation Language (SRL) is straightforward and will allow TRC to offer the services for traceability recovery and reuse without too much development effort.
- Technological level. The SysMLV2 API inherits the architectural style that can be found in the family of ISO STEP standards and OSLC-based services. It aims at providing resources and operations under a REST architectural style (apart from the native connectors) which helps to improve the interoperability of tools at the

communication and syntax levels. The possibility of implementing the SysMLV2 API through the OpenAPI standard will ease the development of clients and it is totally aligned to other efforts in the industry. However, the official developments are not yet ready in the major tool providers so, specific implementations will be still necessary.

- Implementation level. In the case of UC-3, it would be necessary to expose logical models as a service.
- UC-3 specific level. As it has been introduced, the possibility of consuming logical models through an interoperable layer will help to overcome the problems of different serializations and the lack of elements information. The operations of traceability recovery and reuse can then be implemented taking advantage of a fine-grained access to model elements.

Finally, it is important to discuss some implications of creating a federated environment of tools for Systems Engineering processes. These questions have been also raised in the context of the OSLC community and they will be inherited in the SysMLV2 APIs. The first important element is the link management and stability. Usually, an engineering process is managed by a suite of interconnected tools within a PLM (Product Lifecycle Management) or ALM (Application Lifecycle Management). Internally, there is common database or repository that keep all references, so it is easy to back up (archiving) all system artifacts or recover them once the system is finished. In a federated environment of tools, this situation changes completely since services are managed by the specific tools and are exposed through some URIs. The engineering process is not completely self-contained (e.g. tools, methods and system artifacts) which can generate further problems in terms of archiving (e.g. services that are not alive, tools that does not exist anymore, etc.). That is why, an integration of tools through services must carefully consider this situation in the long term. The SysMLV2 API focuses on content-based interoperability representing a big step towards the reuse of models. However, current trends in industry are also focusing on card-based interoperability adding fine-grained metadata to system artifacts to be able to orchestrate the engineering process and to share basic information between participants from a high-level perspective. In this sense, the SysMLV2 API should consider the addition of model identity cards.

On the other hand, the possibility of consuming SysML through a REST (JSON) API (instead of RDF under different serializations) allows integrators to easily access logical models being this approach close to developers' background and know-how. In terms of performance, the consumption of an API can lead us to performance problems when accessing very large models (containing thousands of elements). However, the possibility of querying the API will help to mitigate the potential problems of consuming large set of resources (e.g. filtering some elements out). In conclusion, the SysMLV2 API represents a major step to the adoption of the standard and the integration of tools, although some non-functional aspects such as archiving, link management and performance must be carefully evaluated for each engineering environment.

4.4 MOTIF Model transformation & integration

MOTIF is a new initiative from OMG/MantIS group for manufacturing technology. It stands for Model Transformation and Integration Framework/Facility. This initiative seeks pragmatic solutions for Interchange of information captured in models in an (multidisciplinary) engineering collaboration process. It is intended to define a container for (or a Mock-Up of) different models (SysML, Requirements, Office formats, BPMN, etc) with integration capabilities between models (linking between model elements and traceability).

MOTIF will also provide serialization formats for models archival and will assure compatibility to SysML v2. This OMG initiative could be an appropriate place for promoting the Arrowhead SysML Profile in an international standardization group.

4.5 Express/STEP/ISO 10303

In this chapter we present activities in ISO/TC 184/SC 4, Industrial automation systems and automation. SC 4 is developing a large range of standards. Here we focus on ISO 10303, “Product data representation and exchange”, also known as STEP, “Standard for the Exchange of Product Model Data”.

The STEP standards that are called Application Protocols (AP) are in daily use world-wide, especially ISO 10303-242 (AP242), “Managed model-based 3D engineering” and its predecessors AP203 and AP214. Application native data are converted to, for example, the AP242 data model and are transported to a receiving application in an ISO 10303 formatted data file. Most design engineers use this method today to exchange CAD-data; this is also done in several Arrowhead Tools (AHT) use cases.

The scope of STEP is, however, much wider than only CAD. It covers much of the lifecycle information of a product; see Figure 31Figure 33. AHT use cases represent analysis and operational data in STEP to support predictive maintenance.

The ISO 10303 data model is written in the data modelling language EXPRESS (ISO 10303-11). EXPRESS was developed by NIST in the 1980s and then tailored to support product data modelling by ISO/TC 184/SC 4.

The EXPRESS data models of the STEP APs do not only control file exchange but may also serve as database dictionaries. Such usage allows to collect data from various heterogeneous applications in a single and consistent repository. Machine learning and similar tools may be executed on the resulting cross domain data sets.

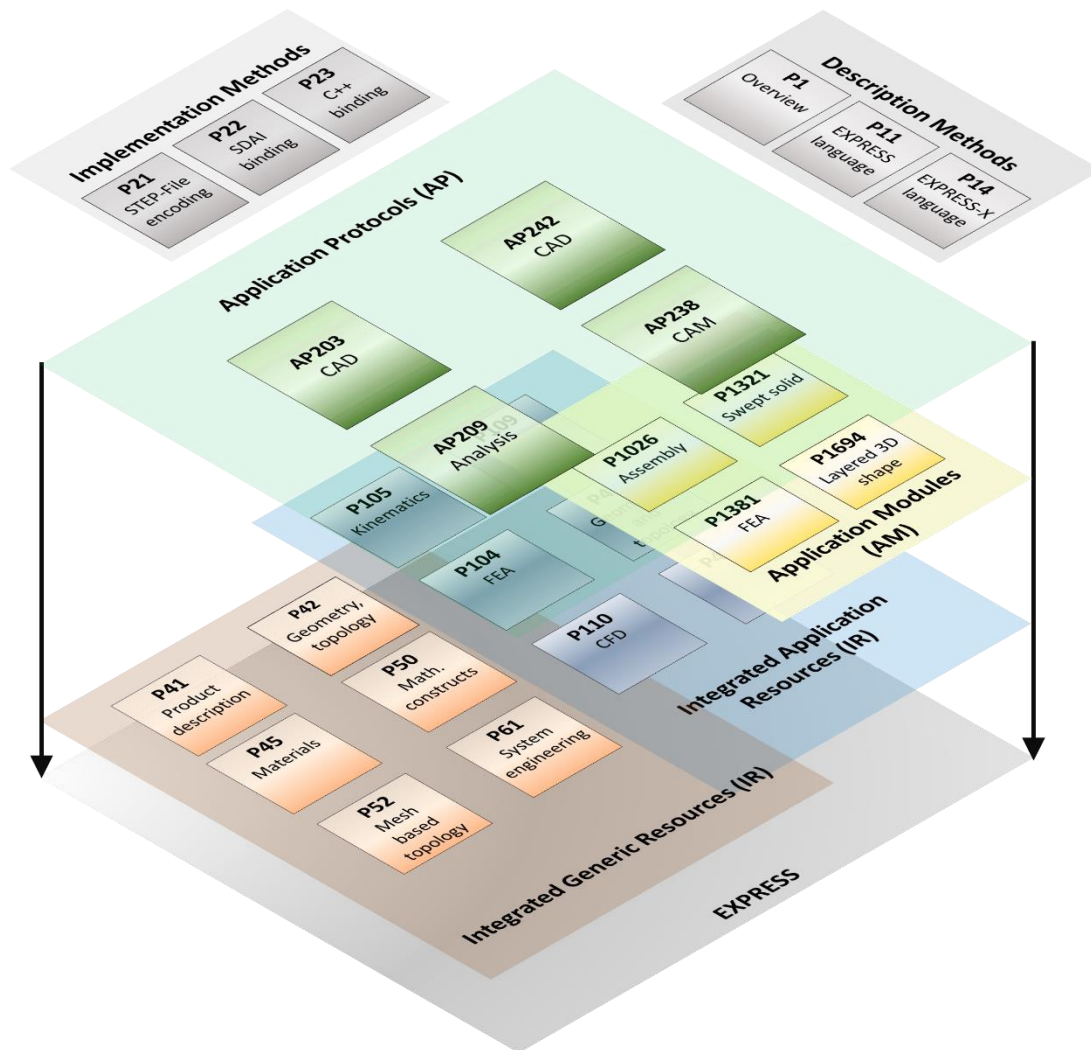


Figure 33: ISO 10303 STEP document structure

Having these STEP usage patterns in mind; we discuss in the following sub-chapters the state-of-the-art of standards developments in SC 4. Focus is on the wide semantics of industrial data covered by STEP and related SC 4 standards. The terminology that is formalized in the ISO 10303 data models – and in other SC 4 documents, naturally overlaps with the terms in engineering domain standards in ISO and in other standardization bodies. The same applies to the ICT items that SC 4 addresses in their role of applying data technology to industrial data. These overlaps may cause conflicts, or they may provide opportunities for collaboration beyond SC 4, to shape standards-based solutions for smart manufacturing and digital twins. This is the reason to report on SC 4 activities here.

4.5.1 Feasibility study for a geometry/topology ontology

WG 12 “STEP Product modelling and resources” in SC 4 is preparing a Technical Report that studies a portion of ISO 10303-42 ”Geometric and topological representation” to provide an equivalent representation in Web Ontology Language (OWL). The work is still on-going and results are not yet publicly available.

The study addresses the following challenges:

1. the ability of OWL profiles to represent the same information with equal precision;
2. the effort required to create the OWL representation;
3. the capability of having geometry and topology ontology to provide a capability for shape and location by being imported into any industrial ontology;
4. the capability of the ontology to support both product and plant geometry and GIS applications;
5. the practicality of the ontology for the representation of large data sets.

ISO 10303-42 (Part 42), currently in its sixth edition, specifies the integrated resources used for the geometric and topological description of the shape representation of products defined in ISO 10303 (STEP), suitable for neutral file exchange, database storage, and as a basis for retention and archiving. The role of Part 42 is critical, as shape representation by geometric description of a product model is at the core of any industrial implementation. The entities from Part 42 are designed to facilitate stable and efficient communication when mapped to a physical file and are in daily use world-wide.

The study considers other attempts that were already done inside and outside of ISO. The mapping of concepts that are defined in the data modelling language of EXPRESS (ISO 10303-11) to a different semantic representation leads into the discussion of formal modelling approaches. This has relevance for Arrowhead as shown in Section 4.6.

4.5.2 ISO 10303 Extended Architecture

ISO 10303 Application Protocols are the end products of ISO 10303, STEP. They include domain specific requirements, a mapping of those to an integrated data model, the resulting target model and implementation requirements with focus on using the AP for data exchange, sharing and archival. For being able to deliver these products ISO/TC 184/SC 4 built the components that an AP consists of according to the ANSI/SPARC three-layer architecture [50]. In STEP, the three macro layers are:

- business layer (business processes, high-level concepts),
- information layer (domain information requirements, mappings, common information structures) and
- implementation layer (programming language bindings to the data model, exchange file formats).

The complexity of developing a data model across all domains and lifecycle phases of all kinds of products required a formal organization also of lower level layer constituents. The following three architectures were introduced over the years, here listed chronologically:

1. Monolithic architecture
2. Modular architecture (see Figure 34)
3. Extended architecture.

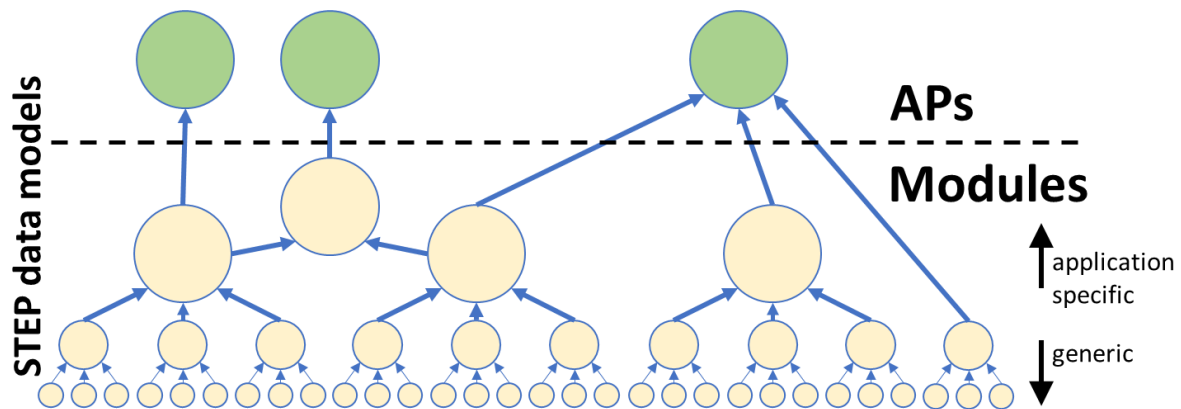


Figure 34: Representation of the principles of the modular STEP information model architecture

The “STEP extended architecture technical report“[51]provides details of these architectures, their justifications, history and characteristics. In this section we will not address these details, but focus on the role of SysML 1.x (ISO/IEC 19514:2017) in the extended architecture to enable a link to the Arrowhead Tools activities with SysML 2 (See Section 4.2).

The extended architecture maintains the above mentioned three layers, but uses SysML on the business layer and the information layer; this gives benefits for the implementation layer.

The business layer is represented on the one hand by one or a group of SysML Activity Models, with each business object being a SysML Activity. On the other hand, there is a similarly structured application data planning model with each business object being modelled as a SysML Block. Such a Block represents an information flow between two Activities and is correspondingly a rather high level information object.

These Blocks are further detailed in the information layer by Application Domain Models that specialize resources of the Core Model; their information element constituents (called Application Objects across all three architectures) are modelled as SysML Blocks. The links between Application Domain and Core Model Blocks are represented by SysML binding connector relationships (displayed on parametric diagrams).

The Application Domain Models are transformed into implementation methods by applying the following description methods, which are currently all in the process of being published as Technical Specifications:

- ISO 10303-15: SysML XMI to XSD transformation
- ISO 10303-16: SysML XMI to EXPRESS transformation
- ISO 10303-17: EXPRESS to SysML CXMI transformation
- ISO 10303-18: SysML XMI to Web services transformation

The SysML mappings to XML (Part 15) and the web-service specification (Part 18) are considered major benefits of the extended architecture compared to the modular one. However, the use of SysML comes with the cost of having to remodel parts of the existing STEP data model that is represented and standardized in EXPRESS to make those available for new domains. One such new Application Domain Models that was recently added describes electric

wire harnesses, for example. Also such new models, for which industry wants to apply the SysML supported implementation methods, need to be mapped to the ISO 10303 data models. Mappings are specified in SysML parametric diagrams between Application Objects of the STEP Modular Architecture, but re-modelled in SysML, and SysML Core Model Blocks. These mappings secure the traceability and interoperability of information elements from the SysML Application Domain Models to the EXPRESS Common Resource Models.

4.5.3 Harmonized terms and vocabulary for industrial data

The data models developed and standardized by SC 4 represent a large collection of industrial data terms. There are several reasons why such terms should be presented in a way that is easier accessible than inside SC 4 data models:

- 1) Industrial users would benefit from knowing that the concepts that they are working with are included in standards for data exchange, sharing and archival.
- 2) Standards developers outside of SC 4 may in their industry domain specific standards deal with the same concepts that SC 4 provides digital representations for.
- 3) Developers of ontologies, industrial or standardized ones, would benefit from the concepts, their relationships and constraints that are already standardized in SC 4 data models. There is obviously a link to 4.5.1 .

In AHT there is the need to refer to manufacturing and infrastructure terms in the MOTIF (See Section 4.4) and SysML modelling efforts (Section 4.2). A harmonization with SC 4 terms may save efforts and increase interoperability.

There is both a SC 4 wide initiative of collecting terms and definitions into a high-level presentation (“AHG 01- Core industrial data set of terms”, ISO/TC 184/SC 4/AHG 1/N30 of 2020-10-12) and an ISO 10303 effort that resulted in a draft ISO 10303-2, “STEP Vocabulary” (ISO/TC 184/SC 4/N3479).

Both list about 100 terms that are intended to provide a comprehensive view of the scope of industrial data. Examples of terms of ISO 10303-2 include Activity, Assembly, Component, Document and Task. The AHG 01 set of terms results from a much more thorough analysis and selection process. It includes the high-level concepts of ISO 10303-2 and completes the selection in a way that may allow its use as an upper ontology.

4.5.4 Use of Reference Data in ISO 10303

The report in [51] defines reference data as follows:

“data that represents information about classes or individuals which are common to many users.

Note 1: Reference Data provides a tailorable structured vocabulary that extends an information model with business specific semantics.

Note 2: This is a generic version of the definition provided in ISO 15926-1:2004, 3.1.18”

The important aspect of reference data in the context of language standards for Arrowhead Tools is the role that reference data may play in data modelling, which is stated in “Note 1”, above: in SC 4 standards reference data are used to specialize the semantics of data models. Having the STEP objective in mind of representing lifecycle data of all kinds of products, it is obvious that this cannot be done in a single data model. New types of products will continuously

be invented; it will not be possible to maintain a standard data model at this pace. SC 4, therefore, embraced the approach of enabling the use of reference data with less specialized data models. Not the data models will need to be updated for new types of products or properties, but one or several reference data libraries (RDL) instead. These RDLs may or may not be subject to standardization; also, different levels of standardization may apply, that is, global, national, industry, project or companywide standardization.

The approach is flexible also from an implementation point of view. The rather stable data model may be used as data dictionary of a database. The dynamic reference data may be accessed by applications at runtime, thus, taking into account most recent updates.

Examples of SC 4 standards with formalized RDLs are ISO 10303-239 (AP239), “Product Lifecycle support (PLCS)”, and ISO 15926-2, “Integration of life-cycle data for process plants including oil and gas production facilities — Part 2: Data model”. AP239 provides its initial set of reference data in OWL, with the EXPRESS entities of the AP239 schema as upper ontology. ISO 15926-2 is accompanied by – a rather large – spreadsheet of reference data, ISO 15926-4 “Initial reference data”.

Several use cases in Arrowhead Tools apply the AP239 solution through the Jotne application EDMtruePLM. Here the OWL formatted reference data are transformed into the EXPRESS schema of ISO 12006-3 “Building construction — Organization of information about construction works — Part 3: Framework for object-oriented information”. Thus, the AP239 schema with its product data is physically in the same database as its reference data. This resolve, among others, the latency of a distributed solution. The richness of the ISO 12006-3 schema allows also to constrain relationships among reference data elements.

The way ahead in SC 4 is to harmonize and to formalize the current handling of reference data – at least within the standards series of ISO 10303 and ISO 15926, but ideally across those - by addressing reference data format, standardization, publication and implementation. Experiences by bodies external to SC 4, including Arrowhead Tools, may accelerate this process.

4.6 OWL and RDF – languages for ontologies

This section provides an overview of the main Semantic Web standards. In addition, we explain where standardization efforts in semantic web are currently being focused and our contributions in this regard.

4.6.1 Standard Semantic technologies

Semantic web encompasses a set of technologies and standards proposed by the World Wide Web Consortium (W3C) used to describe, link, exchange and process data on the Web in a standard and machine-readable way. These standards conform the semantic web technology stack (see Figure 33).

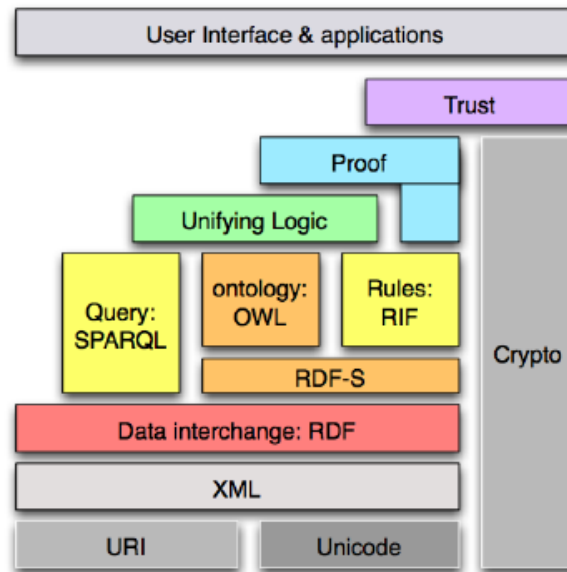


Figure 35: Semantic Web technology stack

- The **Hypertext Transfer Protocol (HTTP) Uniform Resource Identifiers (URIs)** are used to identify uniquely web resources and the HTTP protocol as data retrieval mechanism.
- The standard **Resource Description Framework (RDF)** language [52] is used to specify the data model of the Semantic Web. RDF represents data a set of relations called statements or triples. Each RDF statement has a subject, a predicate and an object. The subject corresponds to an entity or “thing” such as a person or a place. The predicate is a property used to describe an entity, i.e., a person’s name. Objects can be either entities that are subjects of other statements or RDF Literals that describe concrete data values such as string, integer or float values. An RDF statement has the following form:

Subject -> Predicate -> Object

A set of RDF triples constitutes an RDF graph (an example of an RDF graph is shown in Figure 34). The entities or properties of the graph (excepting literals) are identified by URIs. For instance, one of the one of the best-known predicates defined by RDF is *rdf:type* [53], which is used to group the entities together into more general entities. The entities that are grouped into a general entity are considered “instances” of the general entity.

There are different serializations to write RDF statements and graphs. The following are the best known:

- **RDF/XML¹¹**: it uses the XML syntax to serialize RDF.
- **Turtle¹²**: it is a human-readable serialization of RDF.
- **N-Triples¹³**: it is a line-based RDF serialization easy to parse.

¹¹ <https://www.w3.org/TR/rdf-syntax-grammar/>

¹² <https://www.w3.org/TR/turtle/>

¹³ <https://www.w3.org/TR/n-triples/>

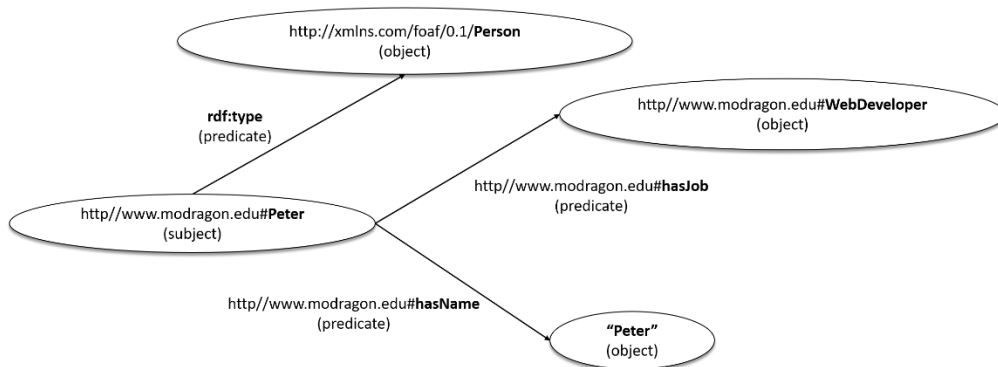


Figure 36: RDF graph example

- The **RDF Vocabulary Definition Language (RDFs)** language¹⁴ is an extension of RDF. RDFS enables the definition of entity and property taxonomies, such as entity or property hierarchies or entity/property domains and ranges. RDFS provides additional vocabularies to describe and relate the data represented in RDF format, so that machines cannot interpret only the explicitly represented data, but also to infer implicit information. Figure 35 shows an example of a basic inference based on the *rdfs:subClassOf* predicate provided by the RDFS language (we omit the full URIs to simplify the diagram).

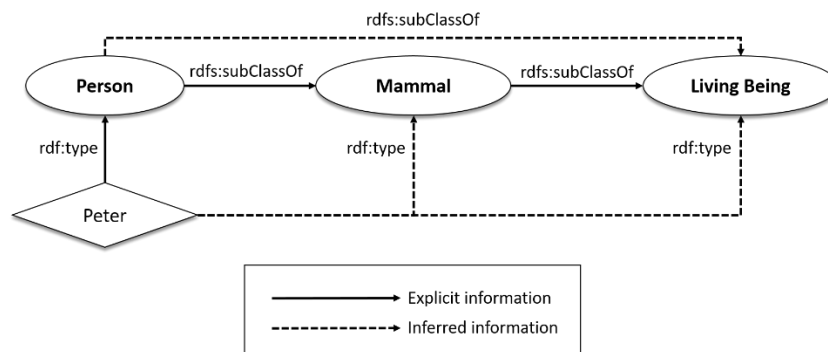


Figure 37: RDFS inference example

The *rdfs:subClassOf* predicate states that all instances that belong to an entity are instances of another. Considering the taxonomy shown in Figure 3, an intelligent agent can infer that all instances of the *Person* entity are instances of the *Living Being* entity. Therefore, the intelligent agent can also infer that *Peter*, apart from being a person, is a mammal and living being.

- The **Web Ontology Language (OWL)** is used to represent complex knowledge about things for applications that need to process the content of information instead of just presenting it to humans. OWL enables establishing more complex relations and semantics than RDF and RDFS language to offer a higher expressiveness when representing web data. These complex relations correspond to rules and axioms. Rules are if-then clauses that allow adding new knowledge to the ontology if certain conditions are met. For example, a rule may define that "If A isMarriedTo B" then this implies "B

¹⁴ <https://www.w3.org/TR/rdf-schema/>

isMarriedTo A". On the other hand, an axiom models a proposition or sentence that is always true. This expressiveness allows intelligent agents to infer new factors about semantically represented web data and improves the data query performance. In summary, OWL provides a basis for creating vocabularies used to describe web data with a high expressiveness. With this knowledge representation, intelligent agents can perform advanced data analysis and reasoning for knowledge extraction and decision-making.

The current version of OWL is OWL-2¹⁵, and it encompasses three sub-languages: OWL Lite, OWL DL and OWL Full. OWL Lite is the one that has lower expressiveness, while OWL Full has the highest.

- The **RDF Query Language (SPARQL)** language [54] is the standard query language of the semantic web and specifically designed to query data represented in RDF, RDFS and OWL language across various systems. SPARQL can be used to express queries across data stored natively as RDF or in other data sources (i.e., XML documents, databases), since these data can be converted to RDF via middleware.

4.6.2 Standardization efforts in domain knowledge representation

As can be seen in previous section, the semantic web provides standard languages to develop semantic ontologies that represent the knowledge of specific data domains. These standards are widely adopted in the research related with the Semantic Web.

Currently, one of the aspects of the Semantic Web where standardization efforts are being made is the domain knowledge representation. Ontologies are usually developed to support specific applications and must satisfy specific knowledge requirements. However, ontologies are developed by different engineers, who represent the knowledge of the same data domains with different vocabularies and level of detail [55], [56]. This knowledge representation diversity is called as *semantic heterogeneity* and hampers the interoperability between the knowledge-based applications and, by extension, the full adoption of ontologies in complex and real scenarios where data exchange between knowledge-based applications is a key requirement [55]. The semantic heterogeneity involves the use of different classes and properties to represent the same concepts and relationships, or the use of different class hierarchies and granularity to represent the same concepts, among other ontology mismatches [57] [58].

The main solution to overcome semantic heterogeneity is to create an ontology that provides a common representation of the knowledge represented by heterogeneous ontologies [55], [56]. These ontologies are known as *global or standard ontologies*, which are a reference to develop ontologies that fit specific application requirements[59]. Global ontologies include common vocabularies to provide a common knowledge representation and a shared understanding of a domain [59],[60]. Thus, the common knowledge of global ontologies can be reused to develop ontologies for different applications [59]. Although local ontologies of each application may not represent the same knowledge, the knowledge they have in common is represented with the same vocabularies. With this common knowledge representation, the applications can

¹⁵ <https://www.w3.org/TR/owl2-overview/>

interoperate to exchange the knowledge represented with common semantic vocabularies within real and complex scenarios [59] (Figure 36). In addition, since the knowledge of the global ontologies is reused to develop application ontologies, the ontology development effort is reduced and engineers can focus more on the application development.

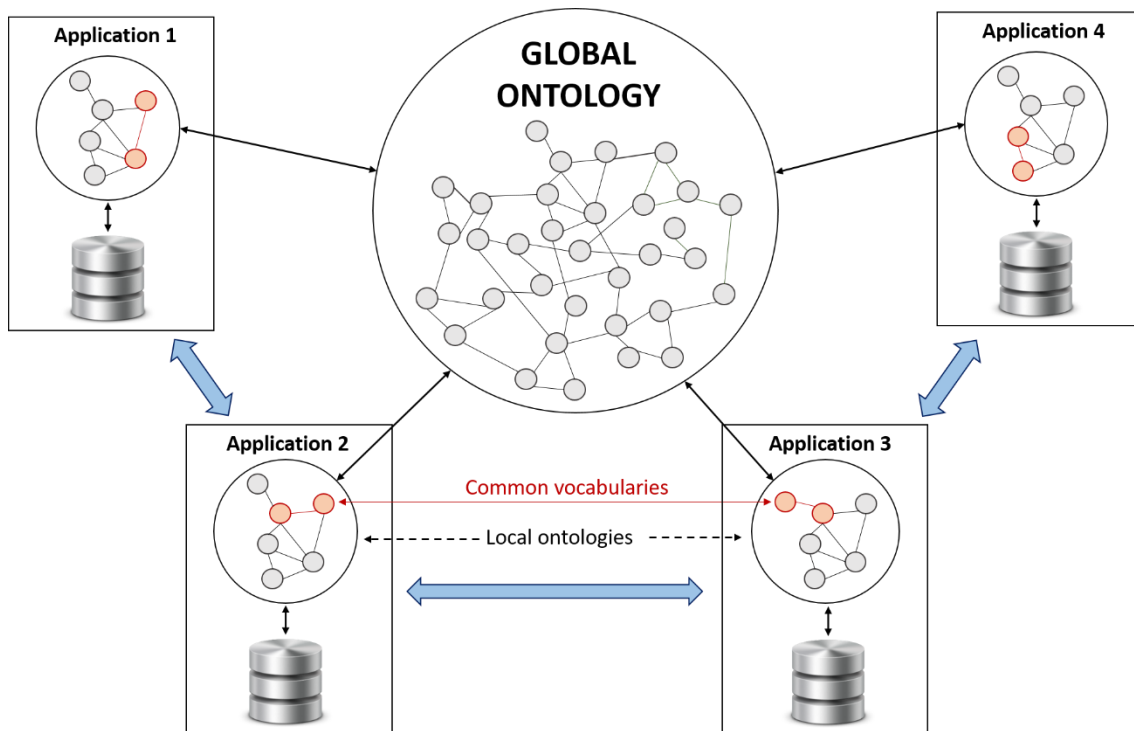


Figure 38: Knowledge exchange with a global ontology

To date, global ontologies have been developed in different domains to provide a common knowledge representation and to enable interoperability between knowledge-based applications. Some examples of global ontologies are the following: SOUPA ontology [61] (developed for the pervasive computing domain), OntoCape ontology [62] (developed for the chemical engineering domain), VSAO ontology [63] (developed for the anatomy domain), MUTO ontology [64] (developed for the web tagging domain), Mobile ontology (for the mobile domain) [65] and IEEE Standard Ontology for Robotics and Automation [66] (developed for the robotics and automation domain).

4.6.3 Our contribution in the standardization of domain knowledge representation

In recent research, we have made two contributions to the standardization of domain knowledge representation:

- The MODDALS methodology [67] (available at <https://innoweb.mondragon.edu/ontologies/MODDALS/index-en.html>). MODDALS guides domain experts and ontology engineers to design layered ontologies. MODDALS takes as reference the knowledge of existing ontologies to define the knowledge of the designed ontology and to classify it into different abstraction layers. Once implemented, the designed ontology can be reused to

develop ontologies for specific applications. Hence, MODDALS enables to design standard ontologies that can be a reference in the domain concerned.

- The DABGEO ontology [68] (available at <http://www.purl.org/dabgeo>). DABGEO provides a common representation of the energy domains represented heterogeneously by the available energy ontologies developed for specific applications. This common knowledge representation enables the creation of interoperable knowledge bases for energy management applications. In addition, it includes links between the vocabularies of the existing energy ontologies to enable interoperability between new and legacy knowledge-based energy management applications. Therefore, DABGEO has the potential to be a standard ontology reused in the energy domain.

4.7 Overview of Industry 4.0 Solutions Based on Semantic Web Technologies

The Semantic Web application to improve the efficiency and resilience of manufacturing processes goes back well before the term Industry 4.0 was coined. During the last two decades, semantic ontologies have been developed to represent manufacturing data from different domains. These ontologies were developed to be the knowledge base of manufacturing management systems focused on optimizing the manufacturing process and improving its resilience.

Since our standardization efforts in domain knowledge representation consider the construction of a global ontology in the industry 4.0 domain, we consider interesting to provide a clear picture of the ontologies available in the field. In the next subsections, we show the state of the art of the ontologies that represent manufacturing domains and ontology-based manufacturing management systems.

4.7.1 Ontologies for Manufacturing Management Systems

Leitao et al. [69] presented the ADACOR (ADaptive holonic COntrol aRchitecture for distributed manufacturing systems) ontology. The ADACOR ontology is the knowledge base of the ADACOR manufacturing control system. This manufacturing system provides a flexible and an agile control of the manufacturing process by reacting to disturbances. The ADACOR ontology provides a standard representation model to the manufacturing control system. Therefore, monitoring and decision control systems can exchange data about the manufacturing process in a standard way, thus improving the interoperability and decision making of the ADACOR manufacturing control system. The ADACOR ontology represents the knowledge about the manufactured products, the product manufacturing process (i.e., operations performed during the manufacturing process, production orders), and disturbances in the manufacturing process (i.e., failures or delays).

Lemaignan et al. [70] presented the MASON (MANufacturing's Semantics ONtology) ontology, an upper ontology that provides a common representation of manufacturing systems. This ontology is aimed to be the knowledge base of a wide variety of manufacturing systems such as manufacturing cost estimation systems or multi-agent manufacturing systems. The MASON ontology represents the knowledge about manufactured products (i.e., material,

manufacturing cost), manufacturing operations (i.e., control, assembly, logistic operations) and manufacturing resources (i.e., human resources, manufacturing tools).

Batres et al. [71] presented an upper ontology based on ISO 15926 standard. The ISO 15926 upper ontology is a top-level ontology developed to support the development of domain ontologies for process engineering systems. The purpose of the ISO 15926 upper ontology is to be a base to develop ontologies that improve the interoperability between applications in chemical process industries. It specifies a conceptual data model for computer representation of technical information about process plants. Based on this standard, the ontology represents the knowledge about top-level concepts (i.e., physical object, spatial location), material phases, activities and events, and physical quantities.

Morbach et al. [62], Natarajan et al. [72] and Muñoz et al. [73] developed ontologies manufacturing process management systems of the chemical engineering field, that is, processes that involve the conversion of chemicals, materials and energy into products in chemical plants, i.e., manufacturing of pharmaceutical products.

Morbach et al. [62], presented the OntoCAPE ontology, a formal ontology specified for computer-aided process engineering (CAPE). The purpose of OntoCAPE is to be reused by a wide variety of process engineering applications. The ontology has been already reused as a knowledge base for different computer-aided applications for design and implementation process engineering [74]. The OntoCAPE ontology represents top-level and domain independent knowledge (i.e., systems, physical quantities, values, units of measure). In addition, it represents the knowledge about material processing and plant operations, materials involved in a chemical process and mathematical models and model building.

Natarajan et al. [62] presented the OntoSafe ontology, which was developed to be the knowledge base of process supervision systems in chemical plants. OntoSafe provides interoperability to intelligent agents that exchange and analyse data to infer the condition of a production plant and detect abnormalities to ensure safe process operation. The OntoSafe ontology relies on the knowledge represented by OntoCAPE ontology. In addition to the knowledge of OntoCAPE, the OntoSafe ontology represents the knowledge about design capacity and limits of plant equipment (i.e., design pressures and temperatures, maximum allowed working pressures), flowsheet information, sensors and the process data they measure, control system description (i.e., control logic, types of controllers required for process modelling), operating procedures (i.e., start-up, shutdown, transition between states) and information on deviations and faults that have previously occurred in the process.

Muñoz et al. [73] presented the BaPrOn ontology (Batch Process Ontology). The purpose of the BaPrOn ontology is to provide a solid and transparent framework for integrating batch-related information of chemical products manufacturing. This ontology provides several benefits to the batch control process: (1) systematic integration of different actors to the control process and (2) common language for better communication about automation opportunities and manufacturing requirements. The knowledge represented by the BaPrOn ontology is based on the ANSI/ISA-88 standard, which provides guidelines for the design and specifications of batch control systems. By following this standard, the BaPrOn ontology represents the knowledge about batch processes, product requirements, product manufacturing process stage, the

equipment required for the production of one or more batches and sensors/actuators that control the batch production process.

Hailemariam et al [75] and Sesen et al. [76] presented ontologies for manufacturing management systems for pharmaceutical plants, within the chemical engineering field. Hailemariam et al [75] presented the POPE (Purdue Ontology for Pharmaceutical Engineering) ontology. POPE is an ontology specifically made for the pharmaceutical process domain within process engineering. The aim of the POPE ontology is to support decision making during the pharmaceutical products and process development, in order to improve the life cycle management of pharmaceutical products. The POPE ontology represents the knowledge about materials, equipment, and production process operation and procedures.

Sesen et al. [76] presented the OntoReg ontology. The aim of OntoReg ontology is to support decision making to in regulatory compliance to enable an automated an intelligent validation of pharmaceutical products against existing regulatory requirements and standards. The OntoReg ontology reuses the knowledge of OntoCAPE and OntoSafe. In addition to these ontologies, OntoReg represents the knowledge about regulatory requirements and the tasks required to address them.

Panetto et al. [77] presented the Onto-PDM (Product-driven ONTOlogy for Product Data Management) ontology. The purpose of the ONTO-PDM ontology is to formalise product technical data to enable interoperability between all the software applications that share information during the physical product lifecycle in a manufacturing environment. For that purpose, ONTO-PDM provides an information structure, as well as expressivity and traceability of the product technical data. The ONTO-PDM ontology represents the knowledge about the lifecycle of manufactured products. This knowledge encompasses product aspects such as product definition, product material, manufacturing equipment, personnel, production capability or production performance.

Lu and Xu [78] presented the ManuService ontology, which is the knowledge base of a cloud manufacturing system that enables an easy mapping between customer requirements and manufacturer resources to enable a fast product manufacturing. The ManuService ontology serves as a generic data model to this system and links data from dynamic service requests with distributed manufacturing resources.

Cheng et al. [79] presented the DeMO ontology (Discrete Manufacturing Ontology). The aim of the DeMO ontology is to support decision making to reconfigure the control software that supervises and coordinates the production process of discrete manufacturing. The DeMO ontology is made up by five ontology modules that represent the following knowledge:

- *Equipment ontology*: equipment used in the domain of production automation and discrete manufacturing.
- *Process ontology*: operations performed by technical equipment.
- *Parameter ontology*: quality of service parameters, intrinsic characteristics of devices and context-dependent information.
- *Product ontology*: conceptual specifications of products.

- *Context ontology*: this ontology links the knowledge of the aforementioned ontologies.

Chhim et al. [80] presented a manufacturing ontology that links key concepts from product manufacturing design and development processes. The purpose of the ontology is to be a knowledge base of systems that improve the product manufacturing process by minimising product disruptions or product failures. The ontology represents the knowledge about product design and production (i.e., stages, functions and failure modes), product design and development team and product customers.

Shi et al [81] presented the OMD ontology (Ontology-Based Manufacturing Description). The purpose of this ontology is to enable manufacturing systems to make automatic adjustments in response to internal manufacturing requirements or external environments to ensure a successful manufacturing process. For that purpose, the ontology provides a support for the analysis and decision of each manufacturing chain. The OMD ontology represents knowledge about manufacturing systems and the services they provide, manufactured products, types of physical equipment and operation behaviours of physical equipment and product characteristics.

Mesmer and Olewnik [82] presented the PMPO ontology (Part-focused Manufacturing Process Ontology). The purpose of this ontology is to enable users with limited knowledge of manufacturing processes to discover potential manufacturers of their product. For that purpose, the ontology is able to provide manufacturing process suggestions to produce the envisioned product based on the information about product manufacturers, the product manufacturing processes they applied to develop previous products and the quality of produced products. The PMPO ontology represents and links the knowledge about types of processes available to produce a product, contact information of manufacturers who apply specific processes to produce a product and alternative processes that can be applied to produce the product.

4.7.2 Ontologies for Industry 4.0 Standards

In addition to the manufacturing ontologies presented in previous section, since the Industry 4.0 term was coined, several ontologies have been developed to represent semantically the Industry 4.0 standards.

Grangel-González et al. [16] presented the RAMI 4.0 vocabulary. The purpose of the RAMI 4.0 ontology is to be a semantic reference model used by intelligent devices to exchange data, thus enabling a self-organized and resilient product manufacturing process. The RAMI 4.0 ontology represents all the relevant knowledge about Cyber Physical Systems (CPSs) that manage the manufacturing process. This knowledge includes device configuration, maintenance and communication with other devices, and device functions. In addition, Grangel-González et al. [83] extended the RAMI 4.0 vocabulary to link it with semantically represented data from other Industry 4.0 standards such as IEC 62264 and eCI@ss.

Kovalenko et al. [84] presented the AutomationML ontology, which is based on the AutomationML standard of Industry 4.0 [85]. The purpose of the AutomationML ontology is to support data analysis activities across the discipline/tool boundaries in Production System Engineering (PSE). The AutomationML ontology represents the knowledge about the

production systems that take part in the manufacturing process such as their function or their role in this process.

4.7.3 Ontology-based manufacturing management systems

Apart from ontologies that represent different manufacturing domains, product manufacturing management systems that use as knowledge base these ontologies have been developed. These systems focus on managing different aspects of the manufacturing process (i.e., optimization of product design accuracy, optimal configuration and scheduling of large-scale manufacturing resources, fast product manufacturing) to improve its efficiency and resilience [78], [86], [87] [88], [89], [90]).

Alsafyand and Vyatkin [90] presented an ontology-based manufacturing system that performs an automatic reconfiguration of manufacturing systems based on changes in manufacturing requirements and environment. The presented manufacturing system decides whether the current environment can support the given manufacturing requirements based on ontologically represented knowledge. This ontology represents the knowledge about the operations involved in the manufacturing process, and the resources required to perform each manufacturing operation.

Cai et al. [89] presented the ManHub system, which manages distributed manufacturing systems. The purpose of this system is to function as a bridge between collaborative enterprises to achieve seamless manufacturing interoperability. The ManHub system is based on an ontology that represents the structural knowledge (i.e., service type, service name and description or associated manufacturing equipment) about heterogeneous manufacturing services. Based on this knowledge representation, the system enables an accurate and automatic retrieval of the required manufacturing services across different enterprises.

Efthymiou et al. [86] presented a knowledge-based framework that supports the early design phases of manufacturing systems. The framework takes as reference the design configurations of previously developed products to optimize and improve the accuracy of the design of new manufacturing systems. The framework is underpinned by an ontology and intelligent agents that identify the best design configurations based on the information of previous products. The ontology that serves as knowledge base to the framework represents the knowledge about products, product design strategies and product key performance indicators.

Zhang et al. [91] presented a cloud manufacturing platform that provides the real-time, accurate, value-added and useful manufacturing information for optimal configuration and scheduling of large-scale manufacturing resources in a cloud manufacturing environment. The ontology that serves as knowledge base to the platform represents the knowledge about manufacturing services: basic attributes (i.e., purchase date), capability attributes (i.e., processing cost), quality attributes (i.e., qualification rate) and real-time status.

Wan et al. [88] presented an intelligent manufacturing system that enables reconfiguration of manufacturing resources to improve equipment intelligence, reduce energy consumption and enhance production efficiency. The presented system is underpinned by an ontology that represents and links the knowledge about manufacturing resources and the processes they perform. This knowledge representation enables the presented manufacturing systems too.

5. Interface and industrial communication standards

The Digital Age, the Society “5.0” or the 4th Industrial revolution represents a challenge and an opportunity to the industry to incorporate new advanced technology elements (IIo,T Cloud Computing, Industrial Communication Protocols,...) that can help in the automation of manual handling tasks, upgrading the workforce activities from operational to innovative activities.

These new automated operational environments augmented with intelligent systems will be engineered using digital twin approaches (automation, modelling and simulation) also shifting the engineering process from a paper-based to a digital asset environment. Then, the systems (e.g. cyber-physical systems) will be equipped with software to build a network of interconnected systems and, in most of cases, with intelligent capabilities. This situation poses some challenges in terms of **engineering non-deterministic features but, more specifically, in operational environments, the communication between software components must be ensured** (e.g. the autonomous car or the factory of the future). Furthermore, the lifecycle of a system is not a set of isolated activities (specially between the design and operation phases), but an evolving lifecycle that must be connected to operational environments to observe the reaction of systems and continuously improve the engineering process.

In this T10.2, we focus on reviewing relevant industrial communication standards, checking the compliance and support of the Arrowhead platform, and observing how the different use cases will be implemented to be able to provide feedback to the standards in two different manners: 1) new applications or uses cases (standards validation) and 2) influence to standardization bodies (standards contribution).

Therefore, T10.2 has been designed to follow a systematic approach to monitor (review) existing standard and mapping the status to the current functionalities of the Arrowhead platform, as well as the needs extracted from the different use cases. The next Figure 38 shows the process that has been followed:

- Identification and selection of relevant industrial communication standards.
- Review and description of the industrial communication standards required by the different use cases (focusing on non-functional aspects).
- Review and description of the industrial communication standards (non-functional aspects) covered by the Arrowhead framework.
- Mapping between QoS (Quality of Service) characteristics of both required/used standards in the use cases and the Arrowhead framework.
- Overlapping analysis and, in the future, definition of the degree of the coverage by the Arrowhead framework.

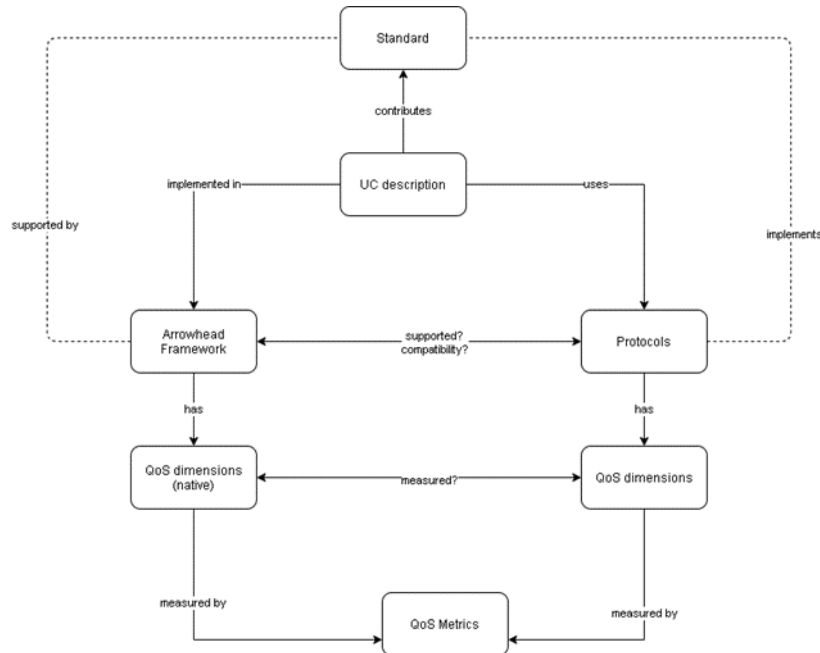


Figure 39: Methodology Process

In this manner, it is possible to systematically measured the support of the framework to the different use cases in terms of industrial communication standards and to establish a roadmap of the standards that should be supported by the framework. The outcomes and the first analysis are described in Section 5.1 and Section 5.2.

In general, the Arrowhead framework [92] [93] already provides components to support non-functional aspects, see Table 5, that are continuously monitoring services and ensuring the SLAs (Service Levels Agreement).

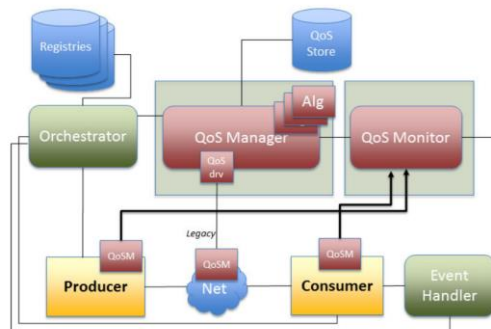


Figure 40: QoS architecture within the Arrowhead framework (Source: [94], [95])

Aspect	Scope	Protocol
Performance	Network	SNMP, Nagios and OpenFlow
Fault Tolerance	Network	
Maintainability	Application Service	
Security (Authentication)	Application Service	

Aspect	Scope	Protocol
Supported protocols	Communication	REST, MTTQ and others
Latency	Network	
Robustness	Network	
Bandwidth	Network	
Timeline guarantee (FTT-SE)	Network	FTT-SE, PROFIBUS, PROFINET and CANopen
Temporal isolation (FTT-SE)	Network	
Data consistency (Communication semantics)	Application Service	
Scalability	Application Service	
Guarantee of service delivery	Application Service	DDS, AMQP / MQTT or XMPP
SLA	Application Service	

Table 4: QoS aspects covered by the Arrowhead Framework based on [94], [95]

Finally, other industrial communication protocol standards worked by partners. specially, NFC and SenML is described in Section 5.3.

5.1 Industrial protocols and characteristics

In the recent decade, various communication and industrial protocols standards have been very intensively studied [96] [97] [98]. A communication protocol can be defined as a set of rules for exchanging information, and, in this review, a list of industrial communication standards and protocols has been extracted considering protocols used for the industrial processes building automation, power-system automation, automatic meter reading or vehicular automation (see Appendix 1).

5.2 Communication Protocols Dimensions within the Arrowhead Framework

5.2.1 Introduction

A communication protocol is, as defined in ¹⁶, *a system of rules that allow two or more entities of a communications system to transmit information. The protocol defines the rules, syntax, semantics and synchronization of communication and possible error recovery methods. Protocols may be implemented by hardware, software, or a combination of both.*

The software part of the protocol is implemented on a framework on top of the operating system of the machine. These models usually have a layered architecture where the functionality of upper layers is built on top of the functionality provided by lower layers. Two of the main frameworks are:

¹⁶ https://en.wikipedia.org/wiki/Communication_protocol

- *OSI (Open Systems Interconnection) model* - This conceptual model, characterises and standardises the communication functions of a system. OSI is a well-known model composed by seven layers; from physical layer (that faces the transmission of data between the device and a physical transmission medium) to application layer that faces how the data will be available to the application that uses the communication.

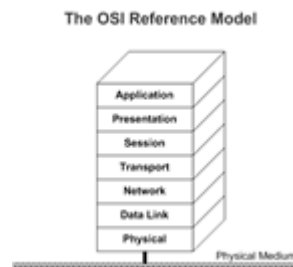


Figure 41: OSI model

Some protocols implement just one layer, such as TCP or UDP, whereas others implement multiple layers, such as OPC-UA or DDS. This characteristic impacts on the complexity of the protocol, the amount of functionality provided, its debuggability, etc.

- *TCP/IP model* – Is the conceptual model used in Internet. This model specifies [5] *how the data should be packetized, addressed, transmitted, routed, and received. This functionality is organized into four abstraction layers, which classify all related protocols according to the scope of networking involved. The lowest layer is the link layer, containing communication methods for data that remains within a single network segment (link); next, the internet layer, providing internetworking between independent networks; next the transport layer, handling host-to-host communication; and finally, the application layer, providing process-to-process data exchange for applications.*

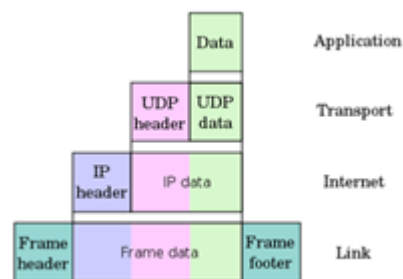


Figure 42: TCP/IP model¹⁷

Communication protocols, whatever it be its base model, must address, define, and specify the following topics:

- *Data formats of messages* – Usually bit strings, that are divided into fields composed by header (name) and payload (data).
- *Address formats* – Identification of addresses of sender and receiver.

¹⁷ https://en.wikipedia.org/wiki/Internet_protocol_suite

- *Routing* – When sender and receiver are not directly connected, intermediate nodes need to know how to route the message.
- *Detection of transmission errors* – Check corruption of message contents (typically with CRC codes).
- *Sequence Control* – When long bit strings are divided in smaller pieces (i.e., packets), those pieces can be lost, delayed, follow different routes, etc. At receiver pieces may arrive out of order or some pieces may be lost. Some mechanisms are needed at receiver to cope with these problems.
- *Flow Control* – If sender sends messages faster than receiver can process, some mechanisms to control the flow may be needed. A typical example here is Apache NiFi (<https://nifi.apache.org/>) to control flow.
- *Etc.*

Arrowhead Framework, following the Service Oriented Architecture philosophy, offers a set of services such as orchestration, authorization, service registry or event handler through API REST. On top of such core services, other domain-oriented services are built. These domain-oriented services main characteristics are:

- provide its functionality via REST,
- make use of event-based communication through the event handler,
- and can make use of other communication protocols.

In the next sections, it will be first described a set of relevant *potential dimensions* of a communication protocol, then, after having a better understanding of communication protocol dimensions, it will be identified *which of them are covered by AHT framework* and finally it will be described how missing dimensions could be provided by other *complementing protocols* like OPC-UA, MQTT, or UMATI, interfacing with AHT REST framework.

5.2.2 Communication Protocols' Potential Dimensions

In the following subsections, a list of potential communication protocol's dimension is described. Later, it will be analysed which of those dimensions are covered by AHT Tools Framework and how protocols like OPC-UA, UMMATI, MQTT, KAFKA can complement AHT in this regard.

5.2.2.1 Connection oriented protocol vs Connectionless oriented protocols

A *connection-oriented* protocol ¹⁸ establishes a communication session or semi-permanent connection and, therefore, ensures that data messages or packets are delivered in the right order. On the opposite side, *connectionless* protocols establish a new connection for every data message or packet and therefore do not guarantee a delivery order, as each packet may be routed in a different communication path.

¹⁸ https://en.wikipedia.org/wiki/Connection-oriented_communication

As an example, one of the most well-known connection-oriented protocols is TCP (Transmission Control Protocol), while an example of connectionless protocol is UDP (User Datagram Protocol).

5.2.2.2 Connection Topology

Topology refers to the layout in which the nodes of the network are physically connected. Topologies basically can be classified into the following types:

- *Star* – The network is organised around a central node (usually called hub) to which all the remaining nodes are connected, in other words, there is no direct link between the nodes. In case a node wants to communicate with another node, it must send the data to the hub, which will resend the data to another node. Advantages of star topology include low cost (although higher than other topologies), easy reconfiguration, easy management, robustness (if link to one node fails, the remaining links are not affected). Its main drawback is that in case of Hub failure, the whole network fails (although hub redundancy is used to cope with this risk).



Figure 43: Star topology.

- *Mesh* – In this case nodes are all connected among them with dedicated point to point connections (therefore each node must have multiple input ports). Obviously dedicated links avoid resource sharing problems, copes well with privacy and security, are very robust because if one link between two nodes fails there are always other possible routes to connect the nodes. However, this topology can be implemented with a very limited number of nodes as the amount of cable and the number of connection ports per node may be costly.

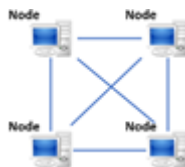


Figure 44: Mesh topology.

- *Bus* – The nodes are connected to a single cable known as *bus*. A transmission of one node can be received by all nodes, it is, is a point to multipoint connection. To avoid collisions and share the bandwidth of the network a mechanism to regulate the use of the bus is required. Installation is easy with a minimum amount of cabling. A failure in the bus stops all transmissions. Privacy and security are not easy to guarantee as all nodes can see messages of all nodes.

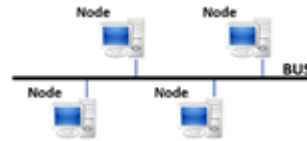


Figure 45: Bus topology.

- *Ring* – Each node has two dedicated point-to-point connections with exactly two neighbouring nodes. The set of all connections forms a close loop. The data is passed along the loop in only one direction, from node to node, until it reaches its destination. A single node, thus, cannot monopolize the network resources. Installation and reconfiguration are very easy as adding/removing nodes implies only two connections. However, if one node fails the whole network may fail.

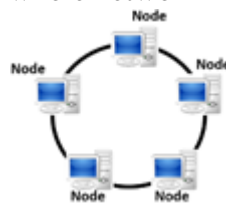


Figure 46: Ring topology.

- *Tree* – Is a general topology in which there is a head at the beginning of the tree in which a single cable starts that may be branched recursively (branches can have subbranches) forming a hierarchical structure without cycles. Nodes are connected to any branches. This type of network is not robust as there is only one route from one node to another. A transmission from one node can be received from any other node so security and privacy are also compromised.

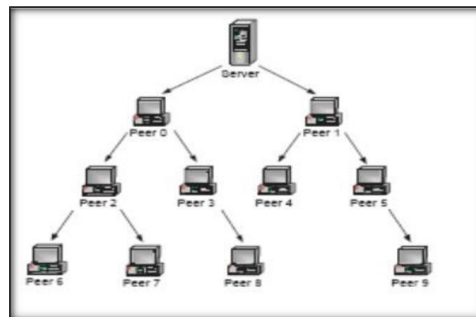


Figure 47: Tree topology.[99]

In summary, protocol's allowed topologies impact directly into other characteristics, such as communication latency, discovery, robustness to failures, security, privacy, etc.

5.2.2.3 Addressing type

A communication protocol connects two or more nodes. To start the communication between the nodes, depending on the addressing type architecture, it is needed to know the address of just some nodes or all the nodes. We can distinguish three types of addressing architectures:

- ***p2p architectures*** (peer-to-peer) – in this type of architectures each node has the same type of capability and responsibility of any other nodes, it is, they are equally privileged and can connect with any other node (i.e. bus, ring, mesh and tree topologies) so the address of the destination node is needed.
- ***client-server architectures*** – in this type of architectures (i.e., star topology) each client node only needs to know the address of a server node (there may be several servers). A client node calls a server node (with known IP and Port), requesting for services. The call blocks the client until it receives an answer from the server (although there may be non-blocking calls for which an answer is received asynchronously via an event).
- ***publisher-subscriber architectures*** – in this type of architecture the publisher nodes send their messages to a privileged node called *broker* under a *topic* category. The subscriber nodes subscribe to a given broker and a given topic. As the publishers *publish* their messages, the subscribed nodes receive the message via events. In this type of architectures, nodes don't know each other, as they only need to know the address of the broker and the topics they want to subscribe to. A typical example of this type of protocol is MQTT (<https://mqtt.org/>).

5.2.2.4 Service Discovery

Wherever it be the addressing type architecture, some protocols provide *automatic ways of discovering* the rest of the nodes, the servers, the brokers, etc. These protocols provide a *Service Discovery* mechanism. Clients can query available services, subscribe to those services, and make requests. In modern cloud-based applications the need of a *Service Discovery* mechanism is crucial because services can change their physical address as the cloud tries to balance loads in different servers. Therefore, actual IP address of services may change dynamically as needed.

There are two main *Service Discovery* patterns: *client-side discovery* and *server-side discovery*.

- ***Client-side discovery pattern*** – The client is responsible for discovering the network locations of available services. The client queries a *Service Registry* (with known address) that is a kind of database of available services. In the answer the client will receive the list of available services and their physical addresses.
- ***Server-side discovery pattern*** – The client makes a request to a service via a load balancer (with known address). It is the load balancer who queries the *Service Registry* and routes the request of the client to the appropriate service.

In both cases the *Service Registry* is a key part of the *Service Discovery* mechanism and is basically a database with the network locations of service instances. Service Registry needs to be up to date and be highly available and complete. Typically, a *Service Registry* consists of a cluster of servers that use a replication protocol to maintain consistency.

5.2.2.5 Security

Secured protocols performs a variety of security-related functions that ensure the security and integrity of data over the network. These protocols are primarily designed to prevent any unauthorized user, application, service, or device from accessing or modifying data in the network. Some examples of these protocols include Secure File Transfer Protocol (SFTP), Secure Hypertext Transfer Protocol (HTTPS) and Secure Socket Layer (SSL).

Typical functions of these protocols are:

- *Authentication & Authorization* – Authenticates the user or application identity. After authentication, the protocol checks permissions and policies and authorizes access to some specific resources.
- *Key Exchange* – Encryption keys are interchanged, as for example in the case of public and private keys.
- *Ciphering* – The contents of the messages are encrypted.
- *Message Integrity* – Mechanism to ensure that the message has not been tampered or altered, as for example, a hash function that combines the bytes in the message with a secret key (i.e., SHA).
- *Etc.*

In case of IoT, security is one of the most important and challenging aspect. As concrete examples:

- MQTT v5.0 is a secure protocol “by design”. Connectivity with the broker is initiated by the client (as MQTT follows publish/subscribe architecture) and therefore clients are protected from attacks as the clients themselves are not addressable via internet. To connect the broker, credential mechanism is provided but also MQTT 5.0 introduces the AUTH protocol packet that allows for more sophisticated authentication and authorization. In addition to this, MQTT V5.0 allows encryption of messages between client and broker using TLS (Transport Layer Security) that is an evolution from SSL.
- OPC-UA connection between the Client and Server make use X.509 certificate standard which define standard public key format and is used by OPC-UA to guarantee:
 - o Communication Integrity – private/public key mechanisms guarantee that the message has not been modified by an attacker and comes from the expected origin.
 - o Communication Encryption – An attacker cannot decrypt the message to see the content.

- o Measures of Trustworthiness – Certificates provide information of what application generated the certificate, when was generated, what the certification can be used for, etc.
- UMATI is built on top of OPC-UA and therefore inherits security mechanisms of OPC-UA.

5.2.2.6 Quality of Service

A protocol defines a Quality of Service (QoS) as the ability of providing and guaranteeing different communication performance to different users or applications, as for example:

- A specific bandwidth.
- A bit error rate.
- A packet loss rate.
- A specific maximum, medium or controlled delay of messages.
- A level of delivery guarantee.
- Etc.

For some application, as for example, real-time streaming such as VoIP, the QoS is a key property to guarantee, because bit rates and controlled delays are necessary for a successful streaming experience.

A protocol that supports QoS may agree with a user or application to provide a given level of QoS, and therefore reserve capacity and resources during a given session for that user or application. In these cases, the protocol may monitor the level of performance and dynamically reschedule priorities. An alternative to complex QoS control mechanism is to provide a *best-effort* network implemented by over-provisioning the capacity of the network, so that the network be able to cope with expected peaks of communication, thus simplifying the protocol underlying mechanisms, but not guaranteeing any level of QoS.

In packet-switched networks, there are several factors that can affect the quality of service. A packet may suffer the following problems during its travel from the sender to the receiver:

- *Low goodput* (measurement of useful data flowing through the network, not to be confused with *throughput* that measures all data flowing through the network, including for example retransmissions, that is a waste of bandwidth) – due to load from multiple users the maximum *goodput* for a given data stream may be too low for real-time multimedia services.
- *Packet loss* – the network fails to deliver packets due to network congestion. The receiving application needs to ask again the retransmission, causing congestion or unacceptable delays in transmission.

- *Errors* – When packets are corrupted with bit errors caused by noise or interferences (typically in Wi-Fi) the receiver needs to ask again for the message.
- *Latency* – if a packet takes too long time to reach its destination (because it is held up in long queues, takes a longer route to avoid congestion, etc) real-time applications like VoIP or games may results unusable.
- *Packet delay variation* – The packets of a given stream of data reach the receiver with different delays. This variation can be absorbed by the receiver (that will re-order the packets) but the overall latency for the stream increases.
- *Out-of-order delivery* – The packets of a given message or stream reaches the receiver in different order than they were sent due to different delays, different positions in the queues of the routers or different paths followed by each packet. The reordering may be done by the protocol itself or by the receiver but, in any case, the overall latency of the stream increases.

5.2.2.7 Delivery guarantee (Reliability)

A communication protocol may or may not guarantee the delivery of the data. This property can also be considered as part of the QoS. Two main groups of protocols may be distinguished:

- *Best-effort-delivery protocols* – In these protocols the delivery is not guaranteed, in other words, it is not guaranteed that the message will reach the destination, and furthermore, the sender will not know if the receiver has received the message. As an example, Internet Protocols are *best-effort* protocols that deliver datagrams between the hosts. For example, in IPv4, datagrams may be lost, delayed, corrupted, or even duplicated.
- *Guaranteed Delivery protocols (Reliable protocols)* – These protocols notify the sender whether the delivery of messages to receivers were successful. These protocols usually have more overhead and are slower than *unreliable best-effort* protocols. As example, Transmission Control Protocols (TCP) provides a guaranteed delivery between hosts, while the User Datagram Protocol (UPD) protocol does not guarantee delivery. UDP is unreliable because it is more oriented to speed (is often used in streaming media where speed is crucial and losing some data is not as important).

As a concrete example of QoS agreement, regarding delivery guarantee, MQTT defines three levels of QoS:

- *At most once (QoS = 0)* – The service just guarantees a *best-effort* delivery but there is no guarantee of delivery (the receiver will not answer acknowledging the reception of

the message). In this case the protocol provides the same guarantee as the underlying TCP protocol.

- *At least once* ($QoS = 1$) – In this case the receiver must answer with an acknowledge. If no acknowledge is received in a given time, the sender may send the message again.
- *Exactly once* ($QoS = 2$) – Using a more sophisticated mechanism of acknowledgement (and more slow mechanism, as there are more acknowledge messages involved) this level of QoS guarantees that each message is received only once by the destination nodes.

5.2.2.8 Scalability

A communication protocol is scalable when an increasement (a) in the participant nodes or (b) the traffic of messages between nodes is gracefully managed without decreasing performance. Many of today's IoT implementations use REST over HTTP to connect clients and servers. REST does not scale well when number of devices or messages per second increases. On the opposite side MQTT is a more scalable protocol, in fact, one of the key goals of MQTT V5.0 is to make easier to host large-scale system in a scalable way.

5.2.2.9 Communication style

Communication between sender and receiver(s) may be asynchronous or synchronous.

- *Asynchronous* – In these protocols the sender sends a message but is not waiting (blocked) for a response. The destination of the sender can be just one node of the network or multiple nodes of the network. In the case of IoT a typical example is MQTT where the sender publishes (sends) a message to a broker node under a given “topic”, and receiver subscribe to the topic and receive the message asynchronously via broker event. This is also called *publish-subscribe* architecture.
- *RPC* – In these protocols the sender calls a remote procedure running in another host and gets waiting (and blocked) for an answer from the receiver of the message. A typical example is a Web Service over http in which the client calling the Web Service is blocked until receiving the answer from the service. This communication style is typical of *client-server* architecture where the caller is the client, and the executor is the server.

5.2.2.10 Transactionality

In information technologies, a transaction is a group of related actions that must be executed as a single action. For example, in the context of data bases, a transaction is a sequence of data storage requests that are treated as a unit. This means that when a commit transaction request is made, the whole data sequences are stored, and in the case of failure, the whole data

sequences storage is undone (roll back). If this property is hold, the database is said to be *transactional*.

A similar concept can be applied to communication protocols. A typical example [100] in event-based architectures is to (1) *update a local database* and later (2) *generate an event* for the services to consume the data. The transaction here must ensure that both operations (1) and (2) are completed, or both operations are not done. In addition to this, involved resources such as databases, messages queues, web services, etc, can be located in the same machines (local transactions) or distributed across several machines (distributed transactions). The latter is a much more complicate case.

5.2.3 Communication Protocols Dimensions covered by AHT framework

This section tries to describe which of the previous dimensions are covered by AHT Framework REST service.

- *Connectionless oriented protocols* – As a REST Web Service, it is a client/server architecture, thus, each interaction with the framework needs a new connection. In case of high of high volumes of data or streaming of data this architecture may cause latency problems.
- *Connection Topology* – AHT framework allows two or more nodes to communicate in two modes:
 - o *peer-to-peer*
 - o *publish/subscribe*.

Therefore, communication topology ranges from a *star* (in which AHT framework acts as a central node) to *mesh* topology as nodes to communicate directly with other nodes (), through the AHT central core services. Although the destination node's address to AHT framework sensors and actuators must be requested first.

- *Addressing type* - AHT framework, as mentioned before, allows nodes to communicate peer-to-peer and publish/subs modes. In addition to this, interaction of each node with AHT follows a client/server pattern.
- *Service Discovery* – AHT framework provides a Service Registry service that can be queried to get to know all services that are registered, or, answer to given name, working with the IP address and port of the service.
- *Security* – AHT core services provide *Authentication & Authorization, Accounting Core System and Encryption Services*. Arrowhead's security relies on HTTPS, X.509 certificates and SSL Certificate Trust Chains that are created by issuing the cloud certificate from the master certificate and the client certificate from the cloud certificate. With other words, the cloud certificate is signed by the master certificate's private key and the client certificate is signed by the cloud certificate's private key which makes the whole chain trustworthy. In addition to this, messages between local clouds that are sent (encrypted) and received (decrypted) by AHT Gateways nodes are certified.

- *Quality of Service* – QoS in a local cloud managed through the QoSManager which produces the *QoSSetup* and a *QoSMonitor* service. The QoSManager is responsible for verifying the feasibility of a QoS request for an orchestration, relating involved parties and monitoring the performance to ensure that QoS requirements are satisfied. The QoS dimensions that can be parameterized and their corresponding allowed parameters are:
 - o *Delay* – End-to-end delay for hard/soft real-time guarantees. Has two parameters:
 - *hardRT*- boolean whether it is hard real-time or soft.
 - *deadline* – integer with maximum delay with end-to-end service consumption in milliseconds.
 - o *Priorisation* – This is applied when no real-time guarantee is possible. Has one parameter:
 - *prio* – integer with the relative priority of the communication flow or service.
 - o *Bandwidth* – Data/message bandwidth and computational bandwidth to accommodate enough service requests. Has two parameters:
 - *requests* – integer with the minimum number of service consumptions that must be satisfied per second.
 - *data* – integer with the minimum number or megabytes that must be produced by the service or transferred by the network active.
 - o *Semantics* – Communication semantics: delivery guarantees and message ordering. Has three parameters:
 - *atLeastOnce* – boolean that when true, at least one copy of the message must be received by its recipient.
 - *atMostOnce* – boolean that when true, no more than one copy of the message must be received by its recipient.
 - *ordered* – boolean that when true, messages in this communication flow must be received in the same order as they were generated.
- Delivery guarantee (Reliability) – AHT framework, as described in the previous QoS parameters allows delivery guarantee (*atLeastOnce* / *atMostOnce*).
- Scalability – AHT framework is a *systems of systems* concept consisting of multiple local cloud architecture at *Edge* level, *Platform* level and *Enterprise* level that allows good scalability without having nonlinear increases in engineering costs and nonlinear increases in possible security problems.
- Communication style (*Asynch* / *Synch*) – both types of communication are possible. As a peer-to-peer system, nodes can communicate synchronously, and as publish/subscribe system, nodes can communicate asynchronously.

5.2.4 Analysis of some Industrial Communication Protocol Complementing AHT framework

As stated in conclusions from deliverable D10.2, partners of Arrowhead Tools project are interested in the interoperability of Arrowhead services with industrial protocols, particularly with OPC-UA (IEC 62541), MQTT, the recent UMATI standard for machine tools, as well as HTTP, WiFi and Ethernet. Next section will pay special attention to those protocols.

5.2.4.1 OPC-UA

It is a M2M communication protocol developed by OPC Foundation ¹⁹. It is an open-source cross-platform software focused on industrial equipment data collection and control. Its main characteristics are:

- Client-Server communication.
- SOA Service Oriented Architecture.
- Offers security functionality for authentication, authorization, integrity, and confidentiality ²⁰.
- Discovery and Global Services.

According to OPC Foundation roadmap ²¹, the following features are “in work” or under consideration for the future (in the latter case, no concrete specification work has been initiated):

- *Features worked on in 2019/2020* (only more relevant to AHT are listed). These are hot topics that the OPC-UA working group addresses and are intended to be added to the current OPC-UA version or to the next full version.
 - o *Provisioning Service* – These Services are designed to allow the security configuration of a Device to be managed over the complete lifecycle of the Device from manufacture to decommissioning. It requires a process to detect counterfeit or modified Devices before they are given access to a sensitive Network.
 - o *Security* – New policies that use Elliptic-curve cryptography (ECC).
 - o *MQTT v5* - MQTT v5 offers features that are needed for routing and filtering by Brokers but also protocol improvements. Currently there are OPC-UA gateways to MQTT, as for example https://www.opc-router.com/4_1-mqtt-client-opc-router-plug-in-en. OPC Router offers gateways from OPC-UA to a variety of systems (see <https://www.opc-router.com/> for details).

¹⁹ <https://opcfoundation.org/about/opc-technologies/opc-ua/>

²⁰ <https://opcconnect.opcfoundation.org/2020/06/exploring-opc-ua-security-concepts/>

²¹ <https://opcfoundation.org/about/opc-technologies/opc-ua/opcua-roadmap/>

- o *Semantic Validation* - This project will add new language elements that allow adding semantic information that today is implied by type or browse names or is written in natural language.

OPC-UA has APIs implementations ²²for the most important languages as C, C++, Java, JavaScript, Rust and Python.

Finally, internally in Arrowhead Tool Project OPC-UA integration to Arrowhead core systems is being worked by LTU. Specifically, paper [101] submitted to MDPI IoT for review.

5.2.4.2 UMATI

The Universal Machine Technology Interface (UMATI) aims at standardizing connectivity and semantics between machinery and software in an easy, secure, and seamless way. UMATI is promoted by the machine builders sector and develops on top of OPC-UA as the global interoperability standard, aiming at:

- Simplifying the effort for machine connection to customer-specific IT infrastructures and software systems.
- Simplifying the effort for machine-to-machine and machine-to-device communication.
- Reducing costs through faster realization of customer specific projects.

To make connectivity between machinery and software easy, secure, and seamless, UMATI creates a community and ecosystem consisting of:

- OPC UA Companion Specifications for a variety of machine builders to define globally applicable semantics for machinery.
- Communication Default Requirements for the implementation of an OPC UA environment such as encryption, authentication, server settings (ports, protocols) to allow plug-and-play connectivity between machines and software.
- Quality Assurance through testing specifications and tools, certification, and serving as ombudsman for supplier-client disputes.

Currently the UMATI has implemented Companion Specifications for the following sectors: Machine-tool, Robotics, Rubber and Plastic, Machine Vision and Scales. As some information items are common to all machinery industry, there is a base Companion Specification for machinery that is under development (first version released in September 2020) that tries to standardize aspects such as:

- Machine identification

²² https://en.wikipedia.org/wiki/OPC_Unified_Architecture

- Component identification
- Finding all Machines in a Server
- Machine state (currently in development)

In addition to that, as an example, UMATI's OPC UA Companion Specification for Machine-Tool (OPC UA 40501) provides the basis for a common machine-tool interface, allowing for example, (a) monitoring the machine, (b) giving an overview of the jobs on it and (c) exchanging information between the machine-tool and software systems like MES, SCADA, ERP, or data analytics systems. OPC UA 40501 is fully endorsed by UMATI community and covers the following uses cases:

- Identify machines of different manufacturers.
- Overview if production is running.
- Overview of parts in a job.
- Overview of runtimes for a job.
- Overview of machine tool state.
- Overview of upcoming manual activities.
- Overview of errors and warnings.
- Providing information for KPI calculations.
- Providing an overview of tool data.

5.2.4.3 MQTT

MQTT is an OASIS standard messaging protocol for IoT. It is a lightweight publish/subscribe messaging transport designed for connecting remote devices to send short messages with minimal bandwidth. Its main features include:

- *Lightweight and efficient* – Clients are very small and (1) can publish messages in topics and (2) subscribe to messages in topics. A special node called *broker* receives and dispatches the messages to topics.
- *Reliable Messages Delivery* – MQTT define three levels of QoS when delivering messages: (0) at most once (no guarantee), (1) at least once, (3) exactly once.
- *Bi-directional Communication* – MQTT allows that the same node can publish messages in a given topic, but also can subscribe to messages from other topics. So, the same node can send and receive messages.
- *Support for Unreliable Networks* – If IoT devices are connected over unreliable networks MQTT supports for persistent sessions reducing the time reconnect the client with the broker.
- *Scalability* – Good scalability and can connect millions of IoT devices.

- *Security* - Provides message encryption (using TLS) and client authentication.

MQTT is one of the most widely used IoT protocols. It is frequently used in combination with the Kafka, as describe in the next section.

Furthermore, MQTT and CoAP support has been added to the mandatory core systems of Eclipse Arrowhead. Planned to be released in May 2021 as part of the 4.4.0 release.

5.2.4.4 MQTT + Kafka

For some real-time applications MQTT alone can fall short, especially when the following requirements are present:

- Real-time data integration and data processing.
- Critical 24/7 deployments where downtimes are not allowed.
- Large-scale processing of events from thousands of users, devices, and machines, as for example:
 - o Track and monitor cars, trucks, fleets, etc in real-time for logistics.
 - o Continuously capture and analyse sensor data from IoT devices such as machines in factories, wind parks, etc.
 - o Monitor patients in hospital to predict changes before emergencies happen.
 - o Etc.

Apache Kafka [13] is the “de facto” open-source standard for distributed event streaming, thus, complements very well MQTT in these situations, because Kafka provides “buffering and persistence” and decouples (with persistence) producers and consumers.

For mission-critical applications Kafka clusters (Kafka runs in cluster pf one or more servers) is highly scalable and fault-tolerant (if one server fails, other servers will take over their work ensuring continuous operation with any loss of data).

5.2.4.5 Gateways to connect OPC UA with MQTT/REST

There are IoT gateways implementations^{23 24}[14,15] to interface OPC-UA with MQTT with REST. These gateways add MQTT and REST interfaces to OPC-UA servers.

²³ <https://www.kepware.com/en-us/products/kepserverex/advanced-plug-ins/iot-gateway/>

²⁴ <https://www.opc-router.com/>

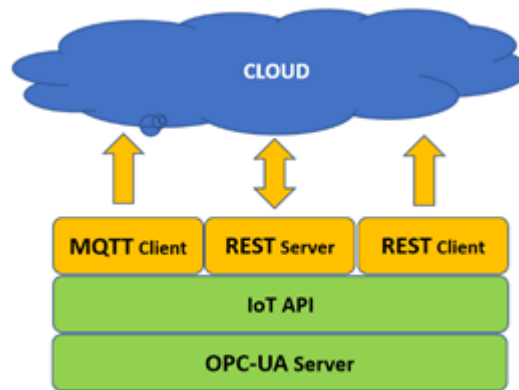


Figure 8 - IoT Gateway from OPC-UA to MQTT and REST

OPC is widely used in industrial automation for communication of machine controllers with plant floor IT systems. Many of the IT applications and systems at floor and plant level (i.e., MES, ERP, or even the cloud) offers API via MQTT and REST, therefore a gateway of OPC-UA opens a number of new possibilities. Interestingly, Arrowhead Tools project offers a REST interface, so this architecture concept fits very well with the project.

Choosing between HTTP REST or MQTT depends on the following aspects:

- For small messages MQTT is more appropriate. In case the messages are composed by large blocks, REST is the right election.
- MQTT is recommended in very small clients, for example small micros fed with batteries, as MQTT has much lower overhead than REST.
- For communication between devices (device to device) MQTT is better, in fact, MQTT was designed for this task.
- MQTT adapts better to unreliable networks by just configuring different levels of QoS.

5.3 Other potential interface and industrial communication standards working through Arrowhead Tools Project

Some highlights of other potential interface and industrial communication standards are described below:

- **Near Field Communication (NFC)**

The NFC Forum is the world's leading standards and advocacy association for Near Field Communication (NFC) technology. One of the major work items of the NFC Forum is the Wireless Charging (WLC) Specification. WLC gives the big opportunity to charge small battery-driven devices using NFC technology. Therefore, the WLC Specification enables wireless charging up to 1 Watt by extending the NFC communication functionality. As now communication and power transfer are possible over a single antenna battery-powered device can be designed in a more cost effective way if recharging is a key.

In the frame of the Arrowhead Tools project, CISC partner drives this work item ahead by acting as the chair of WLC Task Force and the editor of the Wireless Charging Test Specification which might be used in any further certification program. Here, CISC also consult other Work Groups how Wireless Charging can be integrated in the NFC Forum Certification Program. CISC is playing as a contributor a fundamental role over the past 3 years along the publication of the WLC Specification in May 2020.

Within the WLC-TF, CISC is currently focusing on possible improvements of the exiting Reference Equipment to allow a convenient and stable certification. CISC is also contributing to new work items where the NFC Forum works on an extension of the Reference Equipment to cover a certain market need for testing smaller IoT devices, which is also a highlight in the Arrowhead Tools project and the development on WLC solutions.

The NFC Forums announced “The introduction of the wireless charging specification generated the most website traffic in a couple years“. According to this statement it can be conclude that contributing actively to the leading standards and advocacy association the development can be steered in early phase on market needs.

- **SenML standard:**

In the context of UC-16 there is an application of the SenML language for the management of sensors. Although it is not a direct contribution to the standard, it represents a good example of applying some standard and consume through the Arrowhead framework. In this line, future developments will also include the possibility of consuming/producing data and operations under different standards such SysMLV2 (standard under development).

Thus, the Arrowhead platform will provide as built-ins connectors to some of the main standards in the industry for both engineering and operational activities. Specifically SenML standard is supported by the Eclipse Arrowhead core service DataManager, part of the v.4.3.0 release April 2021.

6. Appendixes

Appendix 1 – List of standards and communication protocols

7. References

- [1] J. Qin, Y. Liu, y R. Grosvenor, «A categorical framework of manufacturing for industry 4.0 and beyond», *Procedia Cirp*, vol. 52, pp. 173–178, 2016.
- [2] S. Phuyal, D. Bista, y R. Bista, «Challenges, Opportunities and Future Directions of Smart Manufacturing: A State of Art Review», *Sustain. Futur.*, vol. 2, p. 100023, 2020.
- [3] V. Alcácer y V. Cruz-Machado, «Scanning the industry 4.0: A literature review on technologies for manufacturing systems», *Eng. Sci. Technol. Int. J.*, vol. 22, n.º 3, pp. 899–919, 2019.
- [4] «GMA-Status-Report-RAMI-40-July-2015.pdf». Accedido: mar. 04, 2021. [En línea]. Disponible en:
https://www.zvei.org/fileadmin/user_upload/Presse_und_Medien/Publikationen/2016/januar/GMA_Status_Report_Reference_Architecture_Model_Industrie_4.0__RAMI_4.0_/GMA-Status-Report-RAMI-40-July-2015.pdf.
- [5] Bitkom e.V, VDMA e.V., y ZVEI e.V., «Implementation Strategy Industrie 4.0», p. 104.
- [6] C. Wagner *et al.*, «The role of the Industry 4.0 asset administration shell and the digital twin during the life cycle of a plant», en *2017 22nd IEEE international conference on emerging technologies and factory automation (ETFA)*, 2017, pp. 1–8.
- [7] S. Han, «A review of smart manufacturing reference models based on the skeleton meta-model», *J. Comput. Des. Eng.*, vol. 7, n.º 3, pp. 323–336, 2020.
- [8] E. Barnstedt *et al.*, «Details of the asset administration shell. part 1—the exchange of information between partners in the value chain of industrie 4.0 (version 2.0)», 2018.
- [9] «Details_of-the_Asset_Administration_Shell_Version_2.pdf». Accedido: mar. 04, 2021. [En línea]. Disponible en:
https://www.zvei.org/fileadmin/user_upload/Presse_und_Medien/Publikationen/2020/Januar/Details_of_the_Asset_Administration_Shell_Version_2-0/Details_of-the_Asset_Administration_Shell_Version_2.PDF.
- [10] K. Schweichhart, «Reference architectural model industrie 4.0 (rami 4.0)», *Introd. Available Online Httpswww Plattf. -I40 I*, vol. 40, 2016.
- [11] J. D. Contreras, J. I. Garcia, y J. D. Pastrana, «Developing of Industry 4.0 Applications.», *Int. J. Online Eng.*, vol. 13, n.º 10, 2017.
- [12] I. Grangel-González *et al.*, «The industry 4.0 standards landscape from a semantic integration perspective», en *2017 22nd IEEE International Conference on Emerging Technologies and Factory Automation (ETFA)*, 2017, pp. 1–8.
- [13] J. D. Contreras, J. I. Garcia, y J. D. Pastrana, «Developing of Industry 4.0 Applications.», *Int. J. Online Eng.*, vol. 13, n.º 10, 2017.
- [14] R. Drath, M. Rentschler, y M. Hoffmeister, «The AutomationML Component Description in the context of the Asset Administration Shell», en *2019 24th IEEE International Conference on Emerging Technologies and Factory Automation (ETFA)*, 2019, pp. 1278–1281.
- [15] «Digital-Twin-and-Asset-Administration-Shell-Concepts.pdf». Accedido: mar. 05, 2021. [En línea]. Disponible en: https://www.plattform-i40.de/PI40/Redaktion/EN/Downloads/Publikation/Digital-Twin-and-Asset-Administration-Shell-Concepts.pdf?__blob=publicationFile&v=9.
- [16] I. Grangel-González, L. Halilaj, G. Coskun, S. Auer, D. Collarana, y M. Hoffmeister, «Towards a semantic administrative shell for industry 4.0 components», en

- 2016 IEEE Tenth International Conference on Semantic Computing (ICSC), 2016, pp. 230–237.
- [17] M. Buchheit, «Reference Architecture», p. 58.
- [18] D. M. Pai, «Interoperability between IIC Architecture & Industry 4.0 Reference Architecture for Industrial Assets», *Extern. Doc. White Pap. InfoSys*, 2016.
- [19] M. Yli-Ojanperä, S. Sierla, N. Papakonstantinou, y V. Vyatkin, «Adapting an agile manufacturing concept to the reference architecture model industry 4.0: A survey and case study», *J. Ind. Inf. Integr.*, vol. 15, pp. 147–160, 2019.
- [20] G. Urgese, P. Azzoni, J. van Deventer, J. Delsing, y E. Macii, «An Engineering Process model for managing a digitalised life-cycle of products in the Industry 4.0», en *NOMS 2020-2020 IEEE/IFIP Network Operations and Management Symposium*, 2020, pp. 1–6.
- [21] P. Ghobadi-Bigvand, «An adaptive, context-sensitive, workflow support system for process and automation engineering of production plants», 2018.
- [22] S. Aheleroff, X. Xu, R. Y. Zhong, y Y. Lu, «Digital Twin as a Service (DTaaS) in Industry 4.0: An Architecture Reference Model», *Adv. Eng. Inform.*, vol. 47, p. 101225, 2020.
- [23] M. Hannus, «FIATECH capital projects technology roadmap, ECTP FA PICT», *Available from Httpwww Fiatech Orgtech-Roadmap Accessed25 May 2012*, 2007.
- [24] S. Park, «Development of innovative strategies for the Korean manufacturing industry by use of the Connected Smart Factory (CSF)», *Procedia Comput. Sci.*, vol. 91, pp. 744–750, 2016.
- [25] M. Moghaddam, M. N. Cadavid, C. R. Kenley, y A. V. Deshmukh, «Reference architectures for smart manufacturing: A critical review», *J. Manuf. Syst.*, vol. 49, pp. 215–225, 2018.
- [26] A. Zeid, S. Sundaram, M. Moghaddam, S. Kamarthi, y T. Marion, «Interoperability in smart manufacturing: Research challenges», *Machines*, vol. 7, n.º 2, p. 21, 2019.
- [27] J. Neidig (Siemens AG), A. Orzelski Phoenix Contact GmbH & Co. KG), S. Pollmeier (ESR Pollmeier GmbH), y J. Wende (IBM Deutschland GmbH), «ASSET ADMINISTRATION SHELL READING GUIDE». INDUSTRIE 4.0, nov. 2020, [En línea]. Disponible en: https://www.plattform-i40.de/PI40/Redaktion/EN/Downloads/Publikation/Asset_Administration_Shell_Reading_Guide.pdf?__blob=publicationFile&v=3.
- [28] W. W. Royce, «Managing the development of large software systems: concepts and techniques», en *Proceedings of the 9th international conference on Software Engineering*, 1987, pp. 328–338.
- [29] F. Kevin y M. Harold, «The Relationship of system engineering to the project cycle», *Oslo Nor.*, 1991.
- [30] S. Mathur y S. Malik, «Advancements in the V-Model», *Int. J. Comput. Appl.*, vol. 1, n.º 12, pp. 29–34, 2010.
- [31] A. Cockburn, *Agile software development: the cooperative game*. Pearson Education, 2006.
- [32] S. Balaji y M. S. Murugaiyan, «Waterfall vs. V-Model vs. Agile: A comparative study on SDLC», *Int. J. Inf. Technol. Bus. Manag.*, vol. 2, n.º 1, pp. 26–30, 2012.
- [33] S. W. Popper, S. C. Bankes, R. Callaway, y D. DeLaurentis, «System of systems symposium: Report on a summer conversation», *Potom. Inst. Policy Stud. Arlingt. VA*, vol. 320, 2004.

- [34] B. Scholten, *The road to integration: A guide to applying the ISA-95 standard in manufacturing*. Isa, 2007.
- [35] M. Schleipen, R. Drath, y O. Sauer, «The system-independent data exchange format CAEX for supporting an automatic configuration of a production monitoring and control system», en *2008 IEEE International Symposium on Industrial Electronics*, 2008, pp. 1786–1791.
- [36] M. Schleipen y M. Okon, «The CAEX tool suite-User assistance for the use of standardized plant engineering data exchange», en *2010 IEEE 15th Conference on Emerging Technologies & Factory Automation (ETFA 2010)*, 2010, pp. 1–7.
- [37] R. Y. Zhong, X. Xu, E. Klotz, y S. T. Newman, «Intelligent manufacturing in the context of industry 4.0: a review», *Engineering*, vol. 3, n.º 5, pp. 616–630, 2017.
- [38] O. Carlsson, J. Delsing, F. Arrigucci, A. W. Colombo, T. Bangemann, y P. Nappey, «Migration of industrial process control systems to service-oriented architectures», *Int. J. Comput. Integr. Manuf.*, vol. 31, n.º 2, pp. 175–198, 2018.
- [39] A. Bousdekis, K. Lepenioti, D. Ntalaperas, D. Vergeti, D. Apostolou, y V. Boursinos, «A RAMI 4.0 view of predictive maintenance: software architecture, platform and case study in steel industry», en *International Conference on Advanced Information Systems Engineering*, 2019, pp. 95–106.
- [40] J. Delsing, «Smart city solution engineering, International journal of Smart cities, MDPI», *MDPI Smart Cities*, 2021.
- [41] J. M. Alvarez-Rodríguez, R. Mendieta, V. Moreno, M. Sánchez-Puebla, y J. Llorens, «Semantic Recovery of Traceability Links between System Artifacts», *Int. J. Softw. Eng. Knowl. Eng.*, vol. 30, n.º 10, pp. 1415–1442, 2020.
- [42] H. Lasi, P. Fettke, H. G. Kemper, T. Feld, y M. Hoffmann, *Industry 4.0. Bus Inf Syst Eng* 6: 239–242. 2014.
- [43] D. Seal, «The System Engineering “V” - Is It Still Relevant In the Digital Age?», presentado en Global Product Data Interoperability Summit, 2018, [En línea]. Disponible en: https://gpdisonline.com/wp-content/uploads/2018/09/Boeing-DanielSeal-The_System_-Engineering_V_Is_It_Still_Relevant_In_the_Digital_Age-MBSE-Open.pdf?pdf=Boeing-DanielSeal-The_System_-Engineering_V_Is_It_Still_Relevant_In_the_Digital_Age-MBSE-Open.
- [44] 14:00-17:00, «ISO/DIS 10303-243», *ISO*. <https://www.iso.org/cms/render/live/en/sites/isoorg/contents/data/standard/07/24/72491.html> (accedido mar. 18, 2021).
- [45] D. D. Verma, M. T. McDermott, y K. Pepe, «Research Roadmaps 2019-2020», *Syst. Eng. Res. Cent. SERC Hoboken N. Y. US Tech. Rep. 2020*, p. 12, 2020.
- [46] J. Lu, J. Wang, D. Chen, J. Wang, y M. Törngren, «A service-oriented tool-chain for model-based systems engineering of aero-engines», *IEEE Access*, vol. 6, pp. 50443–50458, 2018.
- [47] J. M. Alvarez-Rodríguez¹, J. Llorens¹, M. Alejandres¹, y J. M. Fuentes², «OSLC-KM: A knowledge management specification for OSLC-based resources», en *INCOSE International Symposium*, 2015, vol. 25, n.º 1, pp. 16–34.
- [48] Object Management Group, Milford, MA, USA, Standard, «Systems Modeling Application Programming Interface (API) and Services, version 2.0», *GitHub*, 2020. <https://github.com/Systems-Modeling/SysML-v2-Release> (accedido mar. 18, 2021).
- [49] S. Huang, V. Gohel, y S. Hsu, «Towards interoperability of UML tools for exchanging high-fidelity diagrams», en *Proceedings of the 25th Annual ACM international Conference on Design of Communication*, 2007, pp. 134–141.

- [50] A. ANSI, «X3/SPARC study group on DBMS, interim report», *SIGMOD FDT Bull*, vol. 7, n.º 2, 1975.
- [51] ISO/TC 184/SC 4/N3403, «“STEP extended architecture technical report“», oct. 30, 2020.
- [52] J. Z. Pan, «Resource description framework», en *Handbook on ontologies*, Springer, 2009, pp. 71–90.
- [53] J. Hebler, M. Fisher, R. Blace, y A. Perez-Lopez, *Semantic web programming*. John Wiley & Sons, 2011.
- [54] J. Pérez, M. Arenas, y C. Gutierrez, «Semantics and Complexity of SPARQL», en *International semantic web conference*, 2006, pp. 30–43.
- [55] M. Maree y M. Belkhatir, «Addressing semantic heterogeneity through multiple knowledge base assisted merging of domain-specific ontologies», *Knowl.-Based Syst.*, vol. 73, pp. 199–211, 2015.
- [56] P. Shvaiko y J. Euzenat, «Ontology matching: state of the art and future challenges», *IEEE Trans. Knowl. Data Eng.*, vol. 25, n.º 1, pp. 158–176, 2011.
- [57] J. De Bruijn, M. Ehrig, C. Feier, F. Martín-Recuerda, F. Scharffe, y M. Weiten, «Ontology mediation, merging and aligning», *Semantic Web Technol.*, pp. 95–113, 2006.
- [58] N. F. Noy y M. A. Musen, «Algorithm and tool for automated ontology merging and alignment», en *Proceedings of the 17th National Conference on Artificial Intelligence (AAAI-00)*. Available as SMI technical report SMI-2000-0831, 2000, vol. 115.
- [59] H. Wache *et al.*, «Ontology-based integration of information—a survey of existing approaches», 2001.
- [60] A. Buccella, A. Cechich, D. Gendarmi, F. Lanubile, G. Semeraro, y A. Colagrossi, «Building a global normalized ontology for integrating geographic data sources», *Comput. Geosci.*, vol. 37, n.º 7, pp. 893–916, 2011.
- [61] H. Chen, F. Perich, T. Finin, y A. Joshi, «Soupa: Standard ontology for ubiquitous and pervasive applications», en *The First Annual International Conference on Mobile and Ubiquitous Systems: Networking and Services, 2004. MOBIQUITOUS 2004.*, 2004, pp. 258–267.
- [62] J. Morbach, A. Wiesner, y W. Marquardt, «OntoCAPE—A (re) usable ontology for computer-aided process engineering», *Comput. Chem. Eng.*, vol. 33, n.º 10, pp. 1546–1556, 2009.
- [63] W. M. Dahdul *et al.*, «A unified anatomy ontology of the vertebrate skeletal system», *PloS One*, vol. 7, n.º 12, p. e51070, 2012.
- [64] S. Lohmann, P. Díaz, y I. Aedo, «MUTO: the modular unified tagging ontology», en *Proceedings of the 7th International Conference on Semantic Systems*, 2011, pp. 95–104.
- [65] C. Villalonga *et al.*, «Mobile ontology: Towards a standardized semantic model for the mobile domain», en *International Conference on Service-Oriented Computing*, 2007, pp. 248–257.
- [66] C. Schlenoff *et al.*, «An IEEE standard ontology for robotics and automation», en *2012 IEEE/RSJ International Conference on Intelligent Robots and Systems*, 2012, pp. 1337–1342.
- [67] J. Cuenca, F. Larrinaga, y E. Curry, «MODDALS methodology for designing layered ontology structures», *Appl. Ontol.*, n.º Preprint, pp. 1–33, 2020.
- [68] J. Cuenca, F. Larrinaga, y E. Curry, «DABGEO: A reusable and usable global energy ontology for the energy domain», *J. Web Semant.*, vol. 61, p. 100550, 2020.

- [69] P. Leitão, «An agile and adaptive holonic architecture for manufacturing control», 2004.
- [70] S. Lemaignan, A. Siadat, J.-Y. Dantan, y A. Semenenko, «MASON: A proposal for an ontology of manufacturing domain», en *IEEE Workshop on Distributed Intelligent Systems: Collective Intelligence and Its Applications (DIS'06)*, 2006, pp. 195–200.
- [71] R. Batres, M. West, D. Leal, D. Price, y Y. Naka, «An upper ontology based on ISO 15926», en *Computer Aided Chemical Engineering*, vol. 20, Elsevier, 2005, pp. 1543–1548.
- [72] S. Natarajan, K. Ghosh, y R. Srinivasan, «An ontology for distributed process supervision of large-scale chemical plants», *Comput. Chem. Eng.*, vol. 46, pp. 124–140, 2012.
- [73] E. Muñoz, A. Espuña, y L. Puigjaner, «Towards an ontological infrastructure for chemical batch process management», *Comput. Chem. Eng.*, vol. 34, n.º 5, pp. 668–682, 2010.
- [74] J. Morbach y W. Marquardt, «Ontology-based integration and management of distributed design data», en *Collaborative and Distributed Chemical Engineering. From Understanding to Substantial Design Process Support*, Springer, 2008, pp. 647–655.
- [75] L. Hailemariam y V. Venkatasubramanian, «Purdue ontology for pharmaceutical engineering: part I. Conceptual framework», *J. Pharm. Innov.*, vol. 5, n.º 3, pp. 88–99, 2010.
- [76] M. B. Sesen, P. Suresh, R. Banares-Alcantara, y V. Venkatasubramanian, «An ontological framework for automated regulatory compliance in pharmaceutical manufacturing», *Comput. Chem. Eng.*, vol. 34, n.º 7, pp. 1155–1169, 2010.
- [77] H. Panetto, M. Dassisti, y A. Tursi, «ONTO-PDM: Product-driven ONTOlogy for Product Data Management interoperability within manufacturing process environment», *Adv. Eng. Inform.*, vol. 26, n.º 2, pp. 334–348, 2012.
- [78] Y. Lu y X. Xu, «A semantic web-based framework for service composition in a cloud manufacturing environment», *J. Manuf. Syst.*, vol. 42, pp. 69–81, 2017.
- [79] H. Cheng, P. Zeng, L. Xue, Z. Shi, P. Wang, y H. Yu, «Manufacturing ontology development based on Industry 4.0 demonstration production line», en *2016 Third International Conference on Trustworthy Systems and their Applications (TSA)*, 2016, pp. 42–47.
- [80] P. Chhim, R. B. Chinnam, y N. Sadawi, «Product design and manufacturing process based ontology for manufacturing knowledge reuse», *J. Intell. Manuf.*, vol. 30, n.º 2, pp. 905–916, 2019.
- [81] Z. Shi, P. Zeng, y H. Yu, «An ontology-based manufacturing description for flexible production», en *2017 2nd International Conference on Advanced Robotics and Mechatronics (ICARM)*, 2017, pp. 362–367.
- [82] L. Mesmer y A. Olewnik, «Enabling supplier discovery through a part-focused manufacturing process ontology», *Int. J. Comput. Integr. Manuf.*, vol. 31, n.º 1, pp. 87–100, 2018.
- [83] I. Grangel-González, L. Halilaj, S. Auer, S. Lohmann, C. Lange, y D. Collarana, «An RDF-based approach for implementing industry 4.0 components with Administration Shells», en *2016 IEEE 21st International Conference on Emerging Technologies and Factory Automation (ETFA)*, 2016, pp. 1–8.
- [84] O. Kovalenko, M. Wimmer, M. Sabou, A. Lüder, F. J. Ekaputra, y S. Biffel, «Modeling automationml: Semantic web technologies vs. model-driven engineering», en

- 2015 *IEEE 20th Conference on Emerging Technologies & Factory Automation (ETFA)*, 2015, pp. 1–4.
- [85] R. Drath, A. Luder, J. Peschke, y L. Hundt, «AutomationML-the glue for seamless automation engineering», en *2008 IEEE International Conference on Emerging Technologies and Factory Automation*, 2008, pp. 616–623.
- [86] K. Efthymiou, K. Sipsas, D. Mourtzis, y G. Chryssolouris, «On knowledge reuse for manufacturing systems design and planning: A semantic technology approach», *CIRP J. Manuf. Sci. Technol.*, vol. 8, pp. 1–11, 2015.
- [87] Y. Zhang, G. Zhang, Y. Liu, y D. Hu, «Research on services encapsulation and virtualization access model of machine for cloud manufacturing», *J. Intell. Manuf.*, vol. 28, n.º 5, pp. 1109–1123, 2017.
- [88] J. Wan, B. Yin, D. Li, A. Celesti, F. Tao, y Q. Hua, «An ontology-based resource reconfiguration method for manufacturing cyber-physical systems», *IEEEASME Trans. Mechatron.*, vol. 23, n.º 6, pp. 2537–2546, 2018.
- [89] M. Cai, W. Y. Zhang, y K. Zhang, «ManuHub: a semantic web system for ontology-based service management in distributed manufacturing environments», *IEEE Trans. Syst. Man Cybern.-Part Syst. Hum.*, vol. 41, n.º 3, pp. 574–582, 2010.
- [90] Y. Alsafi y V. Vyatkin, «Ontology-based reconfiguration agent for intelligent mechatronic systems in flexible manufacturing», *Robot. Comput.-Integr. Manuf.*, vol. 26, n.º 4, pp. 381–391, 2010.
- [91] Y. Zhang, G. Zhang, Y. Liu, y D. Hu, «Research on services encapsulation and virtualization access model of machine for cloud manufacturing», *J. Intell. Manuf.*, vol. 28, n.º 5, pp. 1109–1123, 2017.
- [92] H. Derhamy, J. Eliasson, y J. Delsing, «System of system composition based on decentralized service-oriented architecture», *IEEE Syst. J.*, vol. 13, n.º 4, pp. 3675–3686, 2019.
- [93] P. Varga *et al.*, «Making system of systems interoperable–The core components of the arrowhead framework», *J. Netw. Comput. Appl.*, vol. 81, pp. 85–95, 2017.
- [94] H. Derhamy, J. Eliasson, y J. Delsing, «System of System Composition Based on Decentralized Service-Oriented Architecture», *IEEE Syst. J.*, vol. 13, n.º 4, pp. 3675–3686, dic. 2019, doi: 10.1109/JSYST.2019.2894649.
- [95] P. Varga *et al.*, «Making system of systems interoperable – The core components of the arrowhead framework», *J. Netw. Comput. Appl.*, vol. 81, pp. 85-95, mar. 2017, doi: 10.1016/j.jnca.2016.08.028.
- [96] A. Boukerche, B. Turgut, N. Aydin, M. Z. Ahmad, L. Bölöni, y D. Turgut, «Routing protocols in ad hoc networks: A survey», *Comput. Netw.*, vol. 55, n.º 13, pp. 3032–3080, 2011.
- [97] R.-S. Chang, W.-Y. Chen, y Y.-F. Wen, «Hybrid wireless network protocols», *IEEE Trans. Veh. Technol.*, vol. 52, n.º 4, pp. 1099–1109, 2003.
- [98] P. V. Knudsen y J. Madsen, «Integrating communication protocol selection with hardware/software codesign», *IEEE Trans. Comput.-Aided Des. Integr. Circuits Syst.*, vol. 18, n.º 8, pp. 1077–1095, 1999.
- [99] T. Morais y H. Barbosa, *High Performances Networks Studies*. 2016.
- [100] K. Ch y rakant, «Introduction to Transactions | Baeldung», abr. 25, 2020. <https://www.baeldung.com/transactions-intro> (accedido mar. 19, 2021).
- [101] A. Tripathy, J. van Deventer, y J. Delsing, «Empowering OPC UA with Late Binding», *MDPI IoT*, 2021.



Document title: D10.3 Standardisation report year 2

Version
1.0

Status
Final

Date
2021-04-30

8. Conclusions

The deliverable 10.3 “Standardisation report effort year 2” respond to the need of researching and contributing to the principal standards identified in D10.3 “Standardisation report effort year 2” by partners and uses cases as part of the Task 10.1 ”Language standards” Task 10.2 ” Reference model and methodology standards” and Task 10.3 ” Interface and industrial communication standards”. In addition, with the deployment of D10.2, the Phase 2 established in the WP10 workplan is completed.

Regarding Task 10.1 “Language standards”, standards that are relevant in D10.2 are SysML STEP/ISO 10303, OWL and RDF. These standards respond to the Arrowhead approach which is dependent on adequate means for description, notations and semantics. In the present deliverable, specifically in chapter **Fel! Hittar inte referenskölla.**, it is shown how Arrowhead Tools partners contribute to harmonize the Arrowhead approach with standards in the domain of descriptions. The relationships are two-fold as Arrowhead strive to adhere to dominant standards while also trying to influence the evolution of language standards in the domain of Industry 4.0 in the direction of the Arrowhead approach. The principal actions to be deployed and future works for the next period related to Phase 3 for T10.1 are:

- Section 4.2 showed how a SysML v1 profile for Arrowhead has been established and is now ready for being applied in use-cases. It is also reflected how Arrowhead influence the future SysML v2 through active participation in OMG with clear goals to gain visibility. Within the next year the application of SysML in more use-cases will be deployed with the profile supported by tools Papyrus (of CEA) and MagicDraw (from Dassault, external to our consortium). SysML v2 will reach finalization phase and the Arrowhead influence on SoA constructs should be visible.
- In Section 4.3 the experiment of applying the SysML v2 API for interoperability and flexible tooling on SysML models was presented. Among the next year this will be investigated further as the SysML v2 finalization emerges.
- In Section 4.4 the new initiative MOTIF has been presented seeking for an even wider kind of interchange of models and tooling. The project will influence the initiative by establishing the RFP (Request For Proposals) for the MOTIF standardization process during the next year.
- In Section 4.5, work with data models within ISO focused on ISO 10303 with STEP and EXPRESS have been carried out. This work is related both to the OMG work on SysML, and the work related to ontologies (See Section 4.6). The persistent data models represent a crucial piece of the Arrowhead approach.
- In Sections 4.6 and 4.7 the relationship with ontologies and the many ontologies involved in the Industry 4.0 domain has been explained. Among the next it is expected that the role of ontologies in Arrowhead will be clearer and can contribute to harmonize the heterogeneous landscape that Arrowhead seeks to cover.

Regarding T10.2 “Reference model and methodology standards” the work has been focused on the analysis of Smart Manufacturing References Models. It was aimed to identify the gaps to implement digitalisation and automation solutions considering the whole engineering processes been part of Arrowhead Tools Project, as well as and Industry needs related to digital transformation. The principal actions to be deployed and future works for the next period related to Phase 3 for T10.2 are:

- In Section 3.1 the SMRMs such as RAMI 4.0, AAS, IIRA, “Digital Twin form Manufacturing” and ”A Meta-modelling analysis approach to Smart Manufacturing Reference Models” have been described together with their principal characteristics and standards.
- In Section 3.2 principal gaps and barriers in SMRMs have been explained. SMRMs are mainly focused on the definition of rules for the implementation of I4.0 applications on a high-level point of view. Furthermore, there are few industrial implementations considering SMRM. The analysis carried out highlights that would be necessary to add concepts of IoT, 5G and Digital Twin. In this regard, in [22] a first approach of a Digital Twin Reference Architecture Model is seen. Finally, the most relevant point to consider is the lack of ability to address the Service Oriented Architecture (SOA) and System of System (SoS) domains which are part of the core of Arrowhead Tool Project and considered key in the implementations within Engineering Process.
- In Section 3.3 experiments and contributions considering the analysis, gaps and barriers of SMRMs described in section 3.1 and section 3.2 are presented.

Firstly, industrial use case implementation has been showed using Asset Administration Shell concept. Robotic Arm and Grinding Machine are defined by means of standardized data model implementing the Identification, Documentation and Condition Monitoring sub-models considering AAS specifications. OPC-UA is used to implement data models of industrial devices as a server. The integration module, implemented in NodeRed, considers semantic perspective using GraphDB which facilitates the interaction between among industrial assets. The implementation of the AAS proves the feasibility of this technology in order not only to represent heterogeneous industrial assets and their digital twin, but also to enable the interoperability between those assets in a manufacturing plant. The next period, industrial demonstration will be implemented considering dynamic manufacturing management with AAS for simulating stamping and laser cutting process with different production orders.

Secondly, a tool chain demo is described to show how standardization is useful in practice, the toolchains reference demo developed in WP4 can be used as an example for SoS integration. Thanks to use of standards, a set of most used translators could be predefined and added to the tools catalogue, making the interfacing and interoperability even easier. The next period new versions will be implemented considering standardization.

Regarding Task 10.3 “Interface and industrial communication standards” has served to compile relevant information about 83 industrial communication standards but focusing on full characteristics offered by OPC-UA, MQTT and UMATI. Furthermore, The

dimensions Connection Topology, Addressing type, Service Discovery, Security, Quality of Service, Delivery guarantee (Reliability), Scalability, Communication style (Asynch / Synch) and Transactionality have been analysed. Finally, knowing the Arrowhead tool framework dimensions a mapping between QoS (Quality of Service) characteristics of both required/used standards has been done as well as overlapping analysis and, in the future, definition of the degree of the coverage by the Arrowhead framework. Furthermore, on the one hand Arrowhead Tools allow enterprises to require and provide a standard compliant application and on the other hand Arrowhead framework and communication standards are moving targets, they evolve and new standards are arising continuously. Thereby, a seamless integration between the framework and the already existing or new standards is a key factor. Next steps to bring standardization aspects to Arrowhead covers the explanation of such integration, remarking the extensibility, adaptability and evolution of the solutions built on top of the platform. Finally, NFC and SenML has been taking into account for contributions.

Finally, Phase III, will focus on finishing the standardization contributions as well as considering a standardization Guide and ROI tool even though is not included as part of WP 10 objective. As a result, industrial partners could address the challenge of the standardization and measure the impact of the standards in terms of engineering cost, productivity and efficiency considering an easy roadmap. Besides, it can help for the digital inclusion. That outcome would be based on the work previously developed in the deliverables and added value for all work done before.

9. Revision history

9.1 Contributing and reviewing partners

Contributions	Reviews	Participants	Representing partner
Table of Content	0.1	Michel Iñigo Øystein Haugen José María Alvarez	MON HIOF UC3M
Partner contributions	0.2	WP10 Team	Partners, see Table 1. Standardisation Contacts and Involvement
Integration of Contributions	0.3	Michel Iñigo Agurtzane Labaien Joseba Bilbatua	MON
Review of Contributions	0.4	Michel Iñigo Carolina Mejia	MON
Conclusions, Next Steps, Final Draft	0.5	Michel Iñigo Carolina Mejia Agurtzane Labaien	MON
Review and Feedback	0.6	Jerker Delsing	LTU
Editorial updates	1.0	Mats Johansson	LTU

9.2 Amendments

See section above.

No.	Date	Version	Subject of Amendments	Author

9.3 Quality assurance

No	Date	Version	Approved by
1	2021.04-30	1.0	Jerker Delsing

10. Appendix 1 – List of standards and communication protocols

Area	ID	Name	Organization	Description
Building automation	1-Wire	1-Wire	Dallas Semiconducto	1-Wire is a device communications bus system designed by Dallas Semiconductor Corp. that provides low-speed (16.3 kbit/s[1]) data, signaling, and power over a single conductor.
Automatic meter reading	ANSI C12.18	American National Standard for Utility Industry End Device Data Tables	ANSI	This standard defines a table structure for utility application data to be passed between an end device and a computer. The "end device" is typically an electricity meter, and the "computer" is typically a hand-held device carried by a meter reader, or a meter communication module which is part of an Automatic Meter Reading system.
	ANSI C12.21	American National Standard for Protocol Specification for Telephone Modem Communication	ANSI	
	ANSI C12.22	American National Standard for Protocol Specification for Interfacing to Data	ANSI	ANSI C12.22/IEEE Std 1703 describe a protocol for transporting ANSI C12.19 table data over networks, for the purpose of interoperability among communications modules and meters.

		Communication Networks		
Process automation	AS-i	Actuator Sensor Interface		Is an industrial networking solution used in PLC, DCS and PC-based automation systems. It is designed for connecting simple field I/O devices in discrete manufacturing and process applications using a single two-conductor cable.
Building automation	BACnet	Building Automation and Control (BAC) networks	ISO ANSI ASHRAE	BACnet was designed to allow communication of building automation and control systems for applications such as heating, ventilating, and air-conditioning control (HVAC), lighting control, access control, and fire detection systems and their associated equipment.
Building automation	BatiBUS	BatiBUS	KNX	Was a network protocol for building automation that was introduced in 1989 and has since been succeeded by KNX. It was a relatively simple low-cost protocol that did not rely on dedicated chips
Process automation	BSAP	Bristol Standard Asynchronous/Synchronous Protocol	Emerson	BSAP offers high message security for communication over telephone lines and radio networks by using effective error checking method (16 bit CRC-CCITT) and constantly exchanging the communication statistics.
Process automation, Automobile / Vehicle	CAN bus	Controller Area Network	SAE - BOSCH	Is a robust vehicle bus standard designed to allow microcontrollers and devices to communicate with each other's applications without a host computer.
Building automation	C-Bus	C-Bus	Schneider Electric	Is a communications protocol based on a seven layer OSI

				model for home and building automation that can handle cable lengths up to 1000 metres using Cat-5 cable.
Process automation	CC-Link	CC-Link Open Automation Networks Family	Mitsubishi	The CC-Link Open Automation Networks Family are a group of open industrial networks that enable devices from numerous manufacturers to communicate. They are used in a wide variety of industrial automation applications at the machine, cell and line levels.
Process automation	CIP	Common Industrial Protocol	ODVA	CIP encompasses a comprehensive suite of messages and services for the collection of manufacturing automation applications – control, safety, synchronization, motion, configuration and information.
	ControlNet	ControlNet	ODVA	An open industrial network protocol for industrial automation applications, also known as a fieldbus.
Building automation	DALI	Digital Addressable Lighting Interface	DiiA	DALI is specified by a series of technical standards in IEC 62386. Standards conformance ensures that equipment from different manufacturers will interoperate.
Process automation	DeviceNet	DeviceNet	Rockwell Automation	Is a network protocol used in the automation industry to interconnect control devices for data exchange. It utilizes the Common Industrial Protocol over a Controller Area Network media layer and defines an application layer to cover a range of device profiles.

Process automation	DF-1	DF-1 / DF1		DF-1 / DF1 protocol is an asynchronous byte-oriented protocol that is used to communicate with most Allen Bradley RS232 interface modules. DF1 protocol consists of link layer and application layer formats. DF1 works over half duplex and full duplex modes of communication.
Process automation	DirectNet	Koyo DirectNET	DirectLOGIC	Is used in APS vacuum controls since 1999. It is a master/slave protocol making use of RS-232 or RS-422 physical layers with a baud rate from 300 to 38,400. It is designed to drive a maximum of 90 PLCs on a serial line.
Automatic meter reading	DLMS/IEC 62056	DLMS/IEC 62056		Is a set of standards for electricity metering data exchange by International Electrotechnical Commission.
Power-system automation	DNP3	Distributed Network Protocol 3		Is a set of communications protocols used between components in process automation systems. Its main use is in utilities such as electric, water and SCADA companies.
Building automation	DSI	Display Serial Interface	MIPI	Defines a serial bus and a communication protocol between the host, the source of the image data, and the device which is the destination.
Building automation	Dynet	Dynet	Dynalite	Dynalite components communicate using DyNet. The physical layer consists of a modified RS-485 TIA/EIA-485-A serial bus running along CAT5 cable, blue and blue/white carry the hot and cold signal respectively,

				orange and orange/white carry +12 V DC, green and green/white carry 0 V, Brown and Brown/white are unused.
Process automation	EGD	SCP-ECG	ANSI/AAMI	Is a standard for ECG traces, annotations, and metadata, that specifies the interchange format and a messaging procedure for ECG cart-to-host communication and for retrieval of SCP-ECG records from the host to the ECG cart.
Building automation	EHS	EnOcean	ISO/IEC	EnOcean wireless standard was ratified as the international standard ISO/IEC 14543-3-10.[1] The standard covers the OSI (Open Systems Interconnection) layers 1-3 which are the physical, data link and networking layers.
Process automation	EIB	Ethernet for Control Automation Technology		Is an Ethernet-based fieldbus system, invented by Beckhoff Automation. The protocol is standardized in IEC 61158 and is suitable for both hard and soft real-time computing requirements in automation technology.
Process automation	EnOcean	Ethernet Powerlink	EPSPG	Is a real-time protocol for standard Ethernet. It is an open protocol managed by the Ethernet POWERLINK Standardization Group
Building automation	EtherCAT	European Home Systems	EHTSA	EHTSA was a communication protocol aimed at home appliances control and communication using power line communication (PLC), After merging with two other protocols, it is a part of the KNX standard, which complies with the European Committee for Electrotechnical Standardization (CENELEC)

				norm EN 50090 and has a chance to be a basis for the first open standard for home and building control.
Building automation	Ethernet Powerlink	European Installation Bus		Allows all electrical components to be interconnected through an electrical bus. Every component is able to send commands to other components, no matter where they are. A typical EIB network is made of electrical components such as switches, pulsers, electric motors, electrovalves, contactors, and sensors.
Process automation	FINS	Factory Interface Network Service	Omron	Is a network protocol used by Omron PLCs, over different physical networks like Ethernet, Controller Link, DeviceNet and RS-232C.
Automobile / Vehicle	FlexRay	FlexRay	FlexRay Consortium	Is an automotive network communications protocol developed by the FlexRay Consortium to govern on-board automotive computing. It is designed to be faster and more reliable than CAN and TTP, but it is also more expensive.
Process automation	GE SRTP	Service Request Transport Protocol	GE Intelligent Platforms	The protocol is used over Ethernet almost all GE automation equipment supports the GE-SRTP protocol when equipped with an Ethernet port. Any SRTP client will be capable of reading and writing system memory of any number of remote SRTP capable devices.
Process automation	HART	Highway Addressable	Rosemount Inc.	Its most notable advantage is that it can communicate over legacy 4–20 mA analog

		Remote Transducer		instrumentation current loops, sharing the pair of wires used by the analog-only host systems.
Process automation	HostLink Protocol	HostLink Protocol	Omron	
Process automation	Honeywell SDS	Smart Distributed System	Honeywell	Is an open event-driven protocol used over Controller area network based industrial networks. It is used for a highly reliable Smart device-level network.
	IDB-1394	IDB-1394/IEEE1394	Apple	IDB-1394 Customer Convenience Port (CCP) was the automotive version of the 1394 standard.[
Automobile / Vehicle	IEBus	Inter Equipment Bus	Renesas	Is a communication bus specification "between equipments within a vehicle or a chassis" of Renesas Electronics. It defines OSI model layer 1 and layer 2 specification. IEBus is mainly used for car audio and car navigations, which established de facto standard in Japan, though SAE J1850 is major in United States.
Power-system automation	IEC 60870-5	IEC103	International Electrotechnical Commission	Is a standard for power system control and associated communications. It defines a companion standard that enables interoperability between protection equipment and devices of a control system in a substation.
	IEC 61107	IEC 61107	International Electrotechnical Commission	Or currently IEC 62056-21, was an international standard for a computer protocol to read utility meters. It is designed to operate over any media, including the Internet. A meter sends ASCII (in modes

				A..D) or HDLC (mode E) data to a nearby hand-held unit (HHU) using a serial port.
Power-system automation	IEC 61850	IEC 61850	International Electrotechnical Commission	Is an international standard defining communication protocols for intelligent electronic devices at electrical substations. It is a part of the International Electrotechnical Commission's (IEC) Technical Committee 57[1] reference architecture for electric power systems.
Power-system automation	IEC 62351	IEC 62351	International Electrotechnical Commission	Is a standard developed by WG15 of IEC TC57. This is developed for handling the security of TC 57 series of protocols. The different security objectives include authentication of data transfer through digital signatures, ensuring only authenticated access, prevention of eavesdropping, prevention of playback and spoofing, and intrusion detection.
	INSTEON	INSTEON	Smartlabs	Is a home automation (domotics) technology that enables light switches, lights, thermostats, leak sensors, remote controls, motion sensors, and other electrically powered devices to interoperate through power lines, radio frequency (RF) communications, or both.
Process automation	Interbus	Interbus	Phoenix Contact	Is a serial bus system which transmits data between control systems (e.g., PCs, PLCs, VMEbus computers, robot controllers etc.) and spatially distributed I/O modules that are connected to sensors and actuators

Process automation	IO-Link	IO-Link		Ashort distance, bi-directional, digital, point-to-point, wired (or wireless), industrial communications networking standard (IEC 61131-9) used for connecting digital sensors and actuators to either a type of industrial fieldbus or a type of industrial Ethernet.
Automobile / Vehicle	J1708	SAE J1708	Society of Automotive Engineers	Is a standard used for serial communications between ECUs on a heavy duty vehicle and also between a computer and the vehicle.
	J1939	SAE J1939	Society of Automotive Engineers	Is the vehicle bus recommended practice used for communication and diagnostics among vehicle components.
Building automation	KNX	KNX		Is an open standard for commercial and domestic building automation. KNX devices can manage lighting, blinds and shutters, HVAC, security systems, energy management, audio video, white goods, displays, remote control, etc.
	KWP2000	Keyword Protocol 2000	ISO	Abbreviated KWP2000, is a communications protocol used for on-board vehicle diagnostics systems (OBD). This protocol covers the application layer in the OSI model of computer networking.
Automobile / Vehicle	LIN	Local Interconnect Network	ISO/AWI	Is a serial network protocol used for communication between components in vehicles. The need for a cheap serial network arose as the technologies and the facilities implemented in the car grew

Building automation	LonTalk	LonTalk	ISO/IEC	Is a protocol optimized for control. Originally developed by Echelon Corporation for networking devices over media such as twisted pair, powerlines, fiber optics, and RF
Automatic meter reading	M-Bus	Meter-Bus	EN	Is a European standard (EN 13757-2 physical and link layer, EN 13757-3 application layer) for the remote reading of water, gas or electricity meters.
Process automation	MECHATROLINK	MECHATROLINK	Mechatrolink Members Association (MMA)	Is an open protocol used for industrial automation, originally developed by Yaskawa and presently maintained by Mechatrolink Members Association
Process automation	MelsecNet	MelsecNet	Mitsubishi Electric	This protocol has two variants. MELSECNET/H and its predecessor MELSECNET/10 use high speed and redundant functionality to give deterministic delivery of large data volumes.
Process automation, Building automation	Modbus	Modbus	Schneider Electric	Is a data communications protocol originally published by Modicon (now Schneider Electric) in 1979 for use with its programmable logic controllers (PLCs).
Automobile / Vehicle	MOST	Media Oriented Systems Transport	Microchip Technology	Is a high-speed multimedia network technology optimized by the automotive industry. It can be used for applications inside or outside the car.
	MPEG-1	Moving Picture Experts Group Phase 1	ISO/IEC	Is a standard for lossy compression of video and audio. It is designed to compress VHS-quality raw digital video and CD audio down to about 1.5 Mbit/s

	MQTT	MQ Telemetry Transport	ISO/IEC, OASIS	Is an open OASIS and ISO standard (ISO/IEC 20922) Hightweight, publish-subscribe network protocol that transports messages between devices.
Industrial control system	MT-Connect	MT-Connect	AMT, UCB, GT	Is a manufacturing technical standard to retrieve process information from numerically controlled machine tools.
Building automation	Obix	open Building Information Exchange	OASIS	Is a standard for RESTful Web Services-based interfaces to building control systems. oBIX is about reading and writing data over a network of devices using XML and URIs, within a framework specifically designed for building automation.
	OMG DDS	OMG DDS	OMG	is an Object Management Group (OMG) machine-to-machine (sometimes called middleware or connectivity framework) standard that aims to enable dependable, high-performance, interoperable, real-time, scalable data exchanges using a publish–subscribe pattern.
Industrial control system	OMS	open metering system	Meter-Bus	Stands for a manufacturer- and utilities-independent standardization for Meter-Bus (M-Bus) based communication between utility meters (electricity, gas, water, district heat, heat cost allocators) and systems in the field of smart meters.
Industrial control system	OPC	Open Platform Communicati ons	OPC Foundation	Is a series of standards and specifications for industrial telecommunication. An industrial automation task force developed the original

				standard in 1996 under the name OLE for Process Control
	OPC-UA	OPC Unified Architecture	OPC Foundation	Is a machine to machine communication protocol for industrial automation developed by the OPC Foundation.
Process automation	OpenADR	Open Automated Demand Response		Is a research and standards development effort for energy management led by North American research labs and companies. The typical use is to send information and signals to cause electrical power-using devices to be turned off during periods of high demand.
	Optomux	Optomux	Opto 22	Is a serial (RS-422/RS485) network protocol originally developed by Opto 22 in 1982 which is used for industrial automation applications. Optomux is an ASCII protocol consisting of command messages and response messages containing data from an Optomux unit & contain a message checksum to ensure secure communications.
Process automation	OSGP	Open Smart Grid Protocol	ETSI, ISO/IEC	Is a family of specifications published by the European Telecommunications Standards Institute (ETSI) used in conjunction with the ISO/IEC 14908 control networking standard for smart grid applications.
Process automation	PieP	Process Image Exchange Protocol		Is a very simple Fieldbus protocol used for process automation. It is an application layer protocol developed over TCP/IP. PieP uses method of transferring

				process images between I/O Devices and the PLC which makes the protocol simple to use.
Process automation	Profibus	Process Field Bus	IEC	Is a standard for fieldbus communication in automation technology and was first promoted in 1989 by BMBF (German department of education and research) and then used by Siemens.
Process automation	PROFINET	PROFINET	IEC	Is an industry technical standard for data communication over Industrial Ethernet, designed for collecting data from, and controlling equipment in industrial systems, with a particular strength in delivering data under tight time constraints
Process automation	RAPIDnet	Real-time Automation Protocols for Industrial Ethernet		Is Korea's first Ethernet international standard for real-time data transmission
Process automation	SERCOS III	SERCOS III	IEEE, ISO/IEC	is the third generation of the Sercos interface, a standardized open digital interface for the communication between industrial controls, motion devices, input/output devices (I/O), and Ethernet nodes, such as PCs. Sercos III applies the hard real-time features of the Sercos interface to Ethernet.
Process automation	SERCOS Interface	serial real-time communication system	IEEE, ISO/IEC	Is a globally standardized open digital interface for the communication between industrial controls, motion devices (drives) and input output devices (I/O)

Process automation	Sinec H1	Sinec H1	Siemens	Is an Industrial Ethernet communications protocol that provides the transport layer function widely used in automation and process control applications.
	SynqNet	SynqNet	Danaher Corporation	Is an industrial automation network launched in 2001 by Danaher Corporation for meeting the performance and safety requirements of machine control applications. Synqnet is built over Ethernet link and 100BT physical layer and provides a synchronous connection between various process automation devices
Process automation	TSN	Time-Sensitive Networking		Is a set of standards under development by the Time-Sensitive Networking task group of the IEEE 802.1 working group
Automobile / Vehicle	TTEthernet	Time-Triggered Ethernet (SAE AS6802)		This standard defines a fault-tolerant synchronization strategy for building and maintaining synchronized time in Ethernet networks, and outlines mechanisms required for synchronous time-triggered packet switching for critical integrated applications, IMA and integrated modular architectures
	UAVCAN	Uncomplicated Application-level Vehicular Communication and Networking	Zubax	Is a lightweight protocol designed for reliable intra-vehicle communications using various communications transports, originally destined for CAN bus but targeting various network types in subsequent revisions

	UMATI	Universal machine tool interface	VDW	It enables machine tools and peripherals to connect to customer-specific IT ecosystems – easy, secure, and seamless. umati is an open standard for machine tool users throughout the world. It serves to exploit new potentials for manufacturing of the future.
Automobile / Vehicle	UPB	Universal Powerline Bus	Powerline Control Systems	Is a proprietary software protocol developed by Powerline Control Systems for power-line communication between devices used for home automation.
Building automation	VAN	Vehicle Area Network	ISO	Is a vehicle bus developed by PSA Peugeot Citroën and Renault. It is a serial protocol capable of speeds up to 125 kbit/s and is standardised in ISO 11519-3
Building automation	VSCP	Very Simple Control Protocol		Is a free automation protocol suitable for all sorts of automation task where building- or home-automation is in the main focus.
Building automation	X10		Pico Electronics	Is a protocol for communication among electronic devices used for home automation (domotics). It primarily uses power line wiring for signaling and control, where the signals involve brief radio frequency bursts representing digital information.
Building automation	xAP	xAP		Is an open protocol used for home automation and supports integration of telemetry and control devices primarily within the home. Common communications networks include RS232,

				RS485, Ethernet and wireless. xAP protocol always uses broadcast for sending the messages.
Building automation	ZigBee	ZigBee	IEEE, ISO/IEC	Is an IEEE 802.15.4-based specification for a suite of high-level communication protocols used to create personal area networks with small, low-power digital radios, such as for home automation, medical device data collection, and other low-power low-bandwidth needs, designed for small scale projects which need wireless connection.
	Z-Wave	Z-Wave	Zensys	Is a wireless communications protocol used primarily for home automation. It is a mesh network using low-energy radio waves to communicate from appliance to appliance, allowing for wireless control of residential appliances and other devices