Received: 13 December 2018 | Revised: 5 September 2019 | Accepted: 2 October 2019

DOI: 10.1002/sys.21516

REGULAR PAPER

WILEY

Architecting systems-of-systems and their constituents: A case study applying Industry 4.0 in the construction domain

Jakob Axelsson[1,2] | Joakim Fröberg[2] | Peter Eriksson[3]

[1]School of Innovation, Design and Engineering, Mälardalen University, Västerås, Sweden

[2]RISE SICS, Research Institutes of Sweden, Västerås, Sweden

[3]Blue Institute, Västerås, Sweden

Correspondence
Jakob Axelsson, PhD, School of Innovation, Design and Engineering, Mälardalen University, SE-721 23 Västerås, Sweden.
Email: jakob.axelsson@mdh.se

Funding information
This research was funded by the Swedish Governmental Agency for Innovation Systems (Vinnova), Formas and Sweden's Energy Agency, within their joint program InfraSweden2030 under grant no. 2018-00671, and by Vinnova under grant no. 2018-03244.

Abstract

The development of system-of-systems (SoS) requires a continuous interplay between design decisions on the SoS level and those on the level of its constituent systems (CS), which often preexist and need to be adapted as the SoS evolves. This involves not only preparing the CS to participate in a particular SoS, but also designing the CS architecture to make it easily adaptable to a future SoS context. The problem is in part addressed in an emerging SoS framework in the manufacturing domain called Industry 4.0. It focuses on connected and digitalized production with the ambition of increasing flexibility and efficiency. This paper investigates how Industry 4.0 standards can be used in an SoS context to make CS more flexible and adaptive, and evaluates their usefulness outside manufacturing. The study is based on a case from the construction domain, for which a generic SoS architecture is developed. Several extensions and adaptations of Industry 4.0 are suggested, including specifications of ontologies for missions and workflows.

KEYWORDS

architecture, construction, Industry 4.0, interoperability, system-of-systems

1 | INTRODUCTION

A system-of-systems (SoS) is composed of a number of constituent systems (CS) that have a degree of operational and managerial independence.[1] Often, these CS are preexisting, and need to some extent to be modified to allow them to become members of the SoS. The integration of CS is a well-known challenge, or "pain point," in SoS engineering.[2] This challenge needs to be addressed when designing a new SoS, or evolving an existing one. The issue then becomes how to adapt a particular existing system so that it can become a CS, and this requires by necessity-specific solutions based on the existing design of the system and the SoS. There is therefore a continuous interplay between the architecting of the SoS as a whole, and the architecting of the independent CS. Additionally, as SoS become increasingly common, it is interesting to consider practices to make a system adaptable and interoperable from the start, in order to reduce the effort of later making it a CS of a particular SoS.

In a sense, this question is addressed by the upcoming standards applying the ISO15288 on CS[3] and SoS.[4] However, their focus is on what processes and activities organizations need to put in place in order to successfully support SoS and CS over their life cycles. Equally important is to have good architectural patterns and standardized technical building blocks on these two levels, which can be used in the design of a system to make it flexible enough to easily become a constituent of a yet unknown SoS, and to prepare an existing system to become a constituent of a given SoS. Such patterns and building blocks are the main topic of this paper.

1.1 | SoS in manufacturing

The adoption of SoS is rapidly spreading to more and more domains (even though it is often referred to by other terms than SoS), as the digital transformation of society continues. One area that is making considerable efforts is manufacturing, where cyber-physical systems

is sometimes seen as the fourth industrial revolution, or Industry 4.0 (I4.0), where the first revolution was mechanization powered by steam engines; the second was mass production powered by electrification; and the third was the introduction of computers.[5]

The key principles behind I4.0 as it is being implemented are interconnection, information transparency, decentralized decisions, and technical assistance for human operators.[6] All parts of the manufacturing chain should be connected, have common data models, and provide services to each other.[7] To enable this, a standardization effort is ongoing, which, among other things, covers an architecture framework for I4.0-based SoS, including how a system should be designed to serve as a CS. A relevant question then becomes if these principles are specific to the manufacturing domain, or if they can be applied to SoS in other domains, and this is investigated in the paper.

1.2 | SoS in construction

We are currently in the process of developing a highly extensible SoS framework in the construction domain, with an initial focus on road construction. There are numerous reports, eg, by McKinsey,[8] which indicate a lagging productivity in this domain caused in part by a lack of SoS perspective. Still, as will be shown in Section 7.3, there is only limited research on solutions. An ambition of our work is to lay a foundation for future products and standards in the field that allows increased productivity through connectivity and digital communication. This work is conducted in close collaboration with stakeholders from industry, representing machinery manufacturers, construction companies, and road administration authorities. In that work, we needed to address the question of suitable architectures of both the SoS and the CS. In order to not reinvent solutions that could be found elsewhere, we decided to evaluate if I4.0 concepts were also applicable in our context, and in SoS in general.

An early high-level version of this SoS framework has been presented previously,[9] but is now more mature and is here extended with many details. Also, a part of this work that deals with information representation has been published elsewhere[10] and in more detail, whereas it is in this paper only summarized as part of an architecture description.

1.3 | Contribution and approach

The paper contributes to the state of practice by providing principles and patterns for SoS and CS architecture, with an emphasis on adaptivity and interoperability of CS. The solutions are based on I4.0, but with enhancements that include the introduction of world models for semantic data representation as a foundation for interoperability, and flexible mechanisms for mission and workflow decomposition in a hierarchical SoS. In this way, it also contributes to an in-depth understanding of the relation of I4.0 concepts to SoS in general.

It also proposes concrete solutions for SoS in construction, which is a domain characterized by significantly less repetitive work than manufacturing and with looser managerial control. The paper's primary audiences are both researchers and practitioners, who want to learn more about architectural principles for making CS adaptable and flexible, and on the applicability of design patterns from I4.0 to SoS in domains outside manufacturing.

From a research perspective, the problem has been approached using an industrial case study, in which a concrete architecture description has been derived using a method based on design science.[11] The architecture description has been developed in close interaction with construction specialists in the project's industrial partners to get an in-depth understanding of the processes used and potential improvements. Also, existing descriptions of construction activities, both publicly available and company internal, have been studied. In this way, architectural concerns have been elicited, from which an architecture description was derived.

To validate the architecture description, a software prototype using object-oriented mixed continuous-discrete simulations of the physical construction machines[12,13] has been used as a complement to interactions with the specialists. The purpose of the prototype was to ensure that the parts of the architecture really fit together, and that no major elements were missed.

1.4 | Overview of the paper

In the next section, the key principles of architecture descriptions, I4.0, and ontological information representation are presented, which together act as a frame of reference for this study. In Section 3, the case study from the construction domain is introduced, followed in Section 4 by an analysis of the most important architectural concerns. Section 5 presents the architecture for the case, and thereby also provides an example of how I4.0 can be made concrete. In Section 6, some of the results and experiences are discussed. In Section 7, an overview of previous research in the area is given, and in the final section, the conclusions are summarized together with some indications of future research.

2 | FRAMES OF REFERENCE

The development of the architecture relies on three frames of reference, namely, a standard for architecture descriptions, an architecture framework for I4.0, and concepts for information representation using linked data and ontologies. These are described in the following subsections.

2.1 | Architecture descriptions

This paper describes the architecture of a construction SoS and its constituents. To give a certain rigor to the description, it is based on the ISO42010 standard for architecture descriptions.[14] The standard defines the architecture as the "fundamental concepts or properties of a system in its environment embodied in its elements, relationships, and in the principles of its design and evolution," and the architecture description is a "work product used to express an architecture."

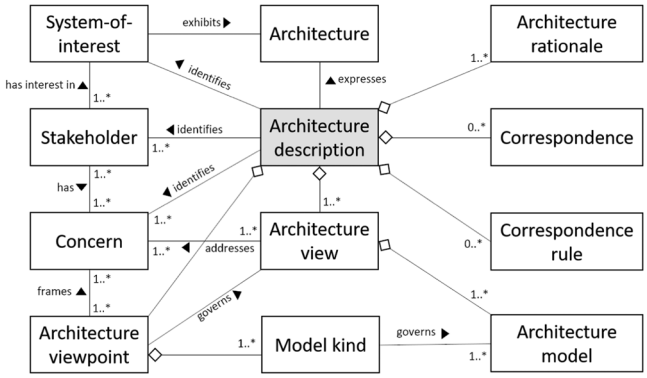


FIGURE 1 UML class diagram illustrating the structure of architecture descriptions according to ISO42010

According to the standard, an architecture description for a particular system-of-interest should include:

- an analysis of *stakeholders* and their *concerns*, which also includes the purpose of the system as well as other expectations or constraints;
- an identification of *correspondences*, or relations, between different elements of the architecture description;
- a presentation of the *rationale* for the decisions made;
- a description of a number of *architecture views* which each express the architecture from the perspective of a specific concern;
- each view is described using the structure of a certain *viewpoint*, where a viewpoint establishes a convention for how to present views related to specific concerns in a given context;
- each view is described using *models* and the models are described using the conventions of a certain *model kind*.

The standard also introduces the concept of an *architecture framework*, which prescribes a common structure or convention for architecture descriptions in a certain domain, and includes among other things a common set of viewpoints. The various elements of an architecture description and their relations are illustrated in Figure 1.

2.2 | Reference architecture model for I4.0

One of the major results of I4.0 is the establishment of a Reference Architecture Model for I4.0 (RAMI4.0). RAMI4.0 has been developed by representatives of a large group of industrial actors in the manufacturing sector in Germany.[15] It can be seen as an architecture framework for the manufacturing domain, and is described in a three-dimensional structure. The three dimensions are: layers; life cycle and value stream; and hierarchy, and these are illustrated in Figure 2. The standard also describes how a CS can be encapsulated in a so-called asset administration shell (AAS), which provides a uniform information interface to I4.0 components. All these concepts are presented in more detail in the remainder of the sections.

2.2.1 | Layers

This dimension represents the architecture of objects, known as assets. An asset can be anything that needs to be represented in the production process, such as physical objects, humans, documents, etc. As a general principle, information exchange should only occur within layers or between adjacent layers. The following layers are included, from top to bottom:

- *Business.* Answers the question what the customer of the product is willing to pay for, and represents the overall business processes as well as orchestrating the functional layer.
- *Functional.* Answers the question what the product is supposed to do, and contains rules and decision-making logic.
- *Information.* Answers what data the product provides, and specifies data persistence and integrity as well as processing events from lower layers into data of different abstraction levels.
- *Communication.* Answers the question how data are to be accessed, and standardizes data formats to be used towards higher layers, as well as control services.
- *Integration.* Answers the question what parts of the product are digitally available in the network, and provides the interface between the physical reality and the IT systems, with information about the assets and events. Also includes Human Machine Interfaces (HMI).
- *Asset.* The physical reality, including humans.

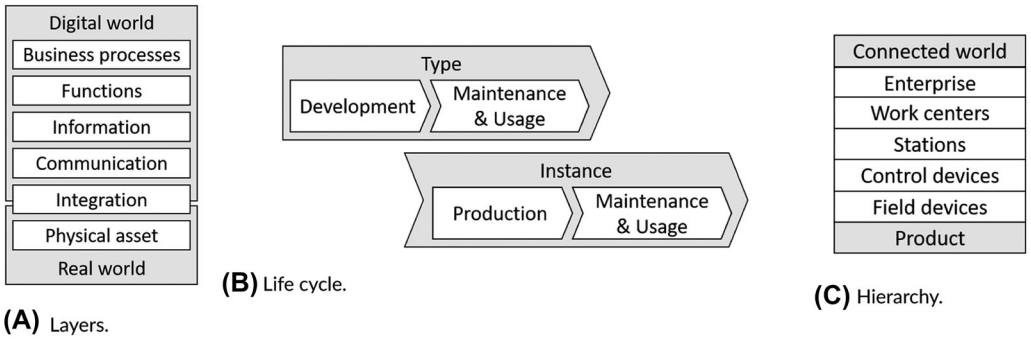


FIGURE 2 Dimensions of Reference Architecture Model Industry 4.0 (RAMI4.0)

2.2.2 | Life cycle and value stream

This dimension describes how assets are described and tracked over their entire lifetime. A clear distinction is made between type and instance, where the former refers to the description (in the form of documents or digital models) of a product, and the latter to the individual physical copies manufactured from that description. The type is thus created during product development, whereas the instances result from the manufacturing process that uses the type description as input. However, there is also a feedback loop that takes data gathered from production into development, allowing improvement.

2.2.3 | Hierarchy

This dimension describes how an asset can be assigned to a technical or organizational hierarchy. It is based on a traditional automation industry hierarchy that includes (from top to bottom): enterprise, work centers, stations, and control and field devices, although it is expected that in the future, this organization is less static than today. It also adds to that at the top a link to the connected world outside the factory, and at the bottom makes the product explicit, since it may nowadays also contain some functionality during the production phase, and not only during the operational phase.

2.2.4 | Asset administration shells

To allow different assets to be combined into a system, I4.0 provides a uniform way of connecting the physical assets into the cyber world. This is called the AAS,[15] and it provides an interface to a number of services that the asset provides. Some services are generic and should be provided by all AAS, including identification, configuration, condition monitoring, events, etc. Others are specific, and include the capabilities or functions that can be performed by the asset. In essence, the AAS is the mechanism provided by I4.0 for turning a system into a CS of the manufacturing SoS.

The AAS consists of a header and a body, where the header contains identifying information about the shell and its asset captured in a manifest, ie, a self-description. The body provides domain-specific submodels, and these are handled through a component manager. I4.0 specifies a standard nomenclature for manufacturing specific services, such as welding, drilling, etc. This is based on a large number of preexisting standards for manufacturing.[16,17] In other domains, such as construction, with less well-defined terminology, the domain-specific taxonomies remain to be formalized and standardized.

There is also an interaction manager responsible for processing communication with other components, and this is done based on a domain-independent basic terminology, which the shell then can map into its own domain-dependent vocabulary.

At a functional level, the asset's functions or capabilities are described in a standardized way. This includes the properties of the function (eg, type, parameters), and its input and output variables. Assets may also be grouped hierarchically, so that a set of assets may be given a common administration shell. The same asset can also have several shells for different purposes.

2.3 | Linked data and ontologies

RAMI4.0 recognizes the need for assigning unique identities to all assets and also to the AAS.[15] However, it does not provide more information about those identifiers. Other documents[18,19] suggest the use of concepts from the Semantic Web initiative.[20]

The main ideas in the Semantic Web are a generic data representation, called linked data, and the use of ontologies for including semantic knowledge. This is captured in the Resource Description Framework (RDF) where each item, which is known as a resource, is assigned a unique identifier that has a syntax similar to a web address. These resources can then be linked to each other through triples, consisting of a subject, a predicate, and an object, where all three elements are resources. Additionally, the objects can also be literal data, such as numbers, strings, etc. As an example, given a resource identifier that represents a particular car and a resource identifier that represents the concept of "speed," it is possible to express the current speed as a triple where the resource identifier representing the car is the subject, the identifier representing the property "speed" is the predicate, and a literal number is the object.

To provide semantics to the linked data, a number of predefined resource identifiers exist that define common concepts from description logic. With these, it is possible to define classes of resources and subclass relations between classes; declare that a resource is an instance of a certain class; define properties of relations, such as the domain and range; etc. This ontological information is also represented as linked data triples, and it allows logical reasoning to derive further information from a set of triples.

A more detailed account on the use of linked data and ontologies in SoS is presented by Axelsson.[10]

3 | OVERVIEW OF CASE STUDY

The construction sector is one of the largest industries in the world, with an annual turnover of around 13% of the global Gross Domestic Product.[8] However, while other industries such as manufacturing have seen productivity improvements in the order of 3.6% per year over the last 20 years, the improvement rate in construction is only about 1% per year,[8] resulting in a huge accumulated difference.

In this case study, we focus on road construction, including quarries and asphalt plants, which is a significant portion of the total construction sector. These road construction activities can be seen as an SoS whose overall structure is an instance of the hierarchical centralized pattern,[21] where the CS at the bottom of the hierarchy are the individual working machines with their operators. These machines have complementing capabilities and need to collaborate. There is, however, a strong element of distributed decision-making, leading to an operational independence of the CS. The first-level SoS is the work site, which could be a quarry, an asphalt plant, or road works. It is quite common that different organizations operate the different sites, introducing an element of managerial independence. Several sites often need to collaborate in order to produce a road, and this creates a second-level

SoS that is the construction project. Here, the individual sites become CS, and one site can at the same time be involved in several projects, such as a quarry that simultaneously delivers to several road construction sites.

The working machine CS have a long lifetime, and can be used for several decades, which means that a typical site contains a mix of machines of different ages. The dynamicity of the SoS, ie, for how long constellations of CS remain stable within the SoS, varies depending on the kind of production. In a quarry site, constellations can be stable for weeks or months, whereas an asphalt laying constellation could be active for a few days or even less.[22]

The different CS are assigned missions from the SoS, and these are decomposed into new missions for the next level, with a feedback flow for reporting progress, etc. Using the common characterization of SoS, the construction activities most closely resemble an acknowledged SoS,[23] in that there are recognized objectives but a larger independence for the CS than in a directed SoS.

In our research, we make a hypothesis that the productivity gap of construction compared to manufacturing is in part due to lack of communication and coordination between the parties involved. Today, this collaboration is handled informally by human communication, and we believe that machine-oriented communication and supporting tools can supplement the human communication in order to improve coordination and identify wastes (ie, activities that do not add value and thus should be eliminated). The system-of-interest in the study is thus a hierarchical cyber-physical SoS that integrates physical CS, such as machines, and provides IT-based decision support to users at all levels. More details about the background of this case have been presented by Axelsson et al.[9]

4 | ARCHITECTURAL CONCERNS

To understand the architectural concerns for the construction SoS, we conducted an analysis that started with the elicitation from the industry participants of user stories, ie, short statements on the following format:

> **As a** <*user role - who?*> **I want to** <*achieve something - what?*> **so that** <*I fulfill a need - why?*>.

Based on this, a list of user types emerged that represent the key stakeholder roles in the SoS. Other nonfunctional needs of the stakeholder roles were also identified, resulting in a set of architectural concerns. In the remainder of this section, we will first briefly describe the stakeholder roles and their needs, and then the nonfunctional architectural concerns are presented.

4.1 | Stakeholder roles and needs

In the analysis of user stories, a number of concrete roles appeared, such as quarry manager and road site manager; hauler operator and excavator operator; etc. These are important to distinguish for the concrete functionality of the system, but to understand the architectural concerns, a generic set of roles is more useful. The stakeholders fall into two broad categories, namely, those related to the bottom-level physical assets and those that relate to a process at one of the higher levels of the hierarchy:

- *Physical asset operator.* The physical asset operators control the assets during production. They primarily need to get information about their current and future missions, including the operating environment (such as maps). They also need status information about other assets with which they collaborate, in order to perform their own tasks as efficiently as possible.
- *Physical asset owner.* The physical asset owners are responsible for asset maintenance and configuration, and hence need current status and forecasts to plan service and reconfiguration to minimize productivity losses.
- *Process operator.* The process operators control the running production of, eg, a site or a project. They need information about their current and future missions; about the current progress of operations to be able to respond to unplanned deviations; and the ability to decompose missions and delegate parts of the mission to CS. They also need to be able to configure the SoS by adding or removing CS.
- *Process quality controller.* The quality controllers' role is to ensure that production fulfills customer requirements, and they therefore need those requirements (captured in the mission) and measurement data from production in order to identify potential deficiencies.
- *Process planner.* The process planners are responsible for planning production to ensure that it can be conducted efficiently. They need forecast information about future missions; capabilities of available assets; and access to operational data to eliminate potential wastes in the current process.

The stakeholders and the system functions they use are illustrated in Figure 3. Although we have only studied the stakeholder roles within the construction SoS, the resulting roles are quite generic and would probably provide a good starting point for stakeholders in other hierarchically central SoS.

4.2 | Nonfunctional concerns

In addition to the functional needs of the stakeholders expressed in user stories, the construction SoS has several nonfunctional architectural concerns, including:

- *Multivendor integration.* Machines from different vendors and of different types must be able to collaborate.
- *Autonomous, remote, and manual operation.* Current construction equipment are mostly manually operated, but there is a strong trend toward automated solutions or remotely controlled machines. The SoS architecture must handle a mix of these types.

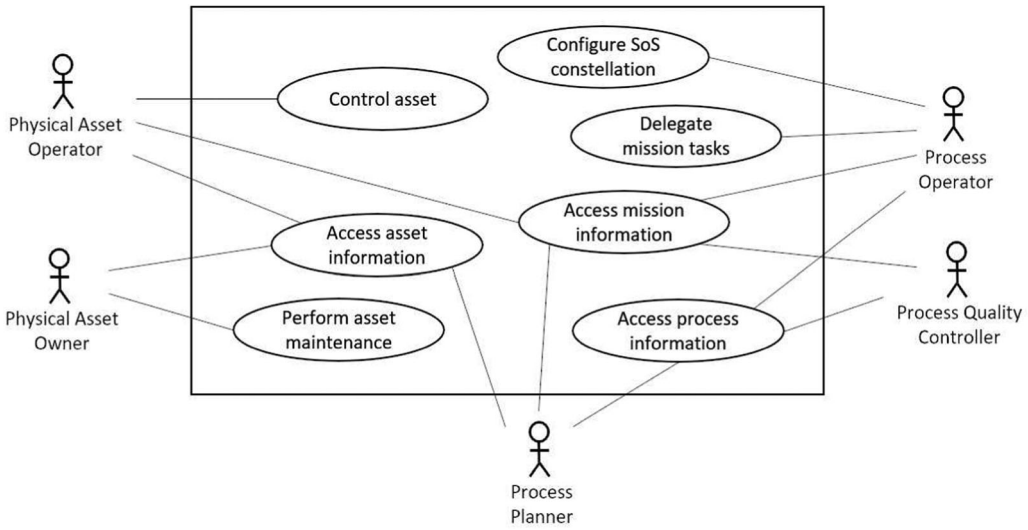


FIGURE 3 UML class diagram illustrating the main stakeholders and system functionality

- *Security.* Participation in an SoS requires some openness, and it must be assured that confidential information of a participant does not become accessible to others, or that operations can be compromised by intruders.
- *Flexibility.* A critical difference between road construction and manufacturing is the continuous changes in the former. The processes have much shorter periods of steady state, which makes process optimization more difficult. This increases the need for up-to-date information, and support for replanning and reconfiguration. The variability between different construction projects is substantial.
- *Robustness.* It cannot be assumed that communication is reliable all the time, since road construction must rely on wireless communication, and the coverage of cellular networks is often poor at construction sites.
- *Stability.* The machines used as CS in a construction SoS have a long lifetime, and the architecture must be stable over time as the SoS evolves.

A particular concern that may appear to be missing from this list is safety, which is obviously an issue when it comes to mobile working machines, and in particular autonomous machines. However, we have not been able to identify additional safety concerns as a consequence of creating an SoS, and therefore have concluded that the already existing safety measures in the individual machines remain sufficient also when part of the SoS.

5 | ARCHITECTURE DESCRIPTION

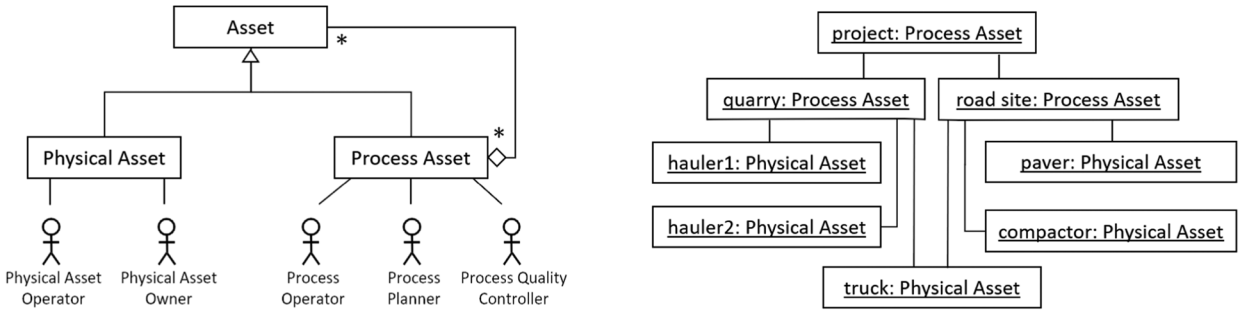
The main parts of the architecture description for the construction SoS will now be introduced. According to ISO42010 (see Section 2.1), this should include a number of views each corresponding to a viewpoint that could come from an architecture framework. However, the RAMI4.0 framework (see Section 2.2) does not provide any viewpoint specifications, and we have therefore chosen to use a set of ad-hoc viewpoints that capture the key architecture elements and concerns as succinctly as possible for our particular application. These viewpoints are described in the first subsection, together with a mapping to the three-dimensional structure of RAMI4.0.

Then, in the remaining subsections, each view corresponding to one of the viewpoints is described by presenting: the conceptual model used, the concerns it covers, and the rationale for the architecture decisions behind it. As is quite normal in architecture descriptions, the views differ a bit in the level of detail provided depending on the number of architectural decisions that underlie them and the number of concerns they address.

5.1 | Viewpoint descriptions

This architecture description uses five different viewpoints to capture the main concerns of the stakeholders:

- *Hierarchy.* Describes the relations between the CS in the SoS, and addresses functional and flexibility concerns. It relates to the hierarchy dimension of RAMI4.0.
- *Asset Integration.* Describes how a physical asset can be given a digital representation, and addresses the concerns about dealing with machines of different types from different vendors, as well as security. This corresponds to the RAMI4.0 layers of integration and asset.
- *Communication.* Shows the information transfer links between AAS, addressing the functional, security, flexibility, robustness, and stability concerns. This corresponds to the communication layer in RAMI4.0, with a focus on the life cycle stage "instance production."
- *Information.* Provides a core ontology of the SoS, addressing concerns for stability, flexibility, and functionality. It relates to the information layer in RAMI4.0.
- *Composition.* Specifies the internal structure of the AAS, dealing with flexibility and stability concerns. It is related to the functional layer of RAMI4.0.



(A) Class diagram.

(B) Object diagram.

FIGURE 4 Hierarchical view depicted as UML models

In all these viewpoints, the model kinds used are Unified Modeling Language (UML) class or object diagrams, complemented with text.

5.2 | Hierarchical view

The first view illustrates the hierarchical relations between the CS in the SoS. This hierarchy is a replication of the existing dominance hierarchy used in construction applications, and the difference is only that the communication is today mostly oral and therefore very slow and sporadic, whereas the proposed architecture provides the means of more efficient continuous coordination through digital communication.

5.2.1 | Model

Figure 4A illustrates, through a UML class diagram, that we have chosen a generic hierarchy, where we differ between the physical assets at the bottom of the hierarchy, and the process assets at higher levels. We have thus not chosen to give specific names to each level in the hierarchy, as RAMI4.0 does. Instead, a recursive relation is used where a process asset, which is a kind of asset, consists of an arbitrary number of other assets. The diagram also shows how the actors mentioned in Section 4.1 interface to the different asset types.

Figure 4B shows an example instance at a given point in time using a UML object diagram. Here, the highest hierarchical level is a project, which is a process asset. It consists of two other process assets, namely, a quarry and a road site. In the quarry, two different haulers represent the physical assets, and at the road site, the physical assets are a paver and a compactor. Note that there is also a truck, which is a physical asset that is part of both the quarry and the road site, since it is used for transporting material between the two sites. The structure is thus not a strict hierarchy, but rather a polyhierarchy.

5.2.2 | Concerns addressed

The hierarchical view addresses the functional concerns related to communication between the hierarchical levels. This includes the needs of physical asset operators to get information about missions; of processes operators to get missions, decompose them to the next level assets, and to configure the constellations in the hierarchy by adding or removing assets; of process planners to access capabilities and data of assets; and of process quality controllers to retrieve mission and measurement data from operations.

The view also addresses the flexibility concern, in that it allows a large variation in how the hierarchy is configured, and provides the ability to change it over time as the processes evolve.

5.2.3 | Rationale

In the manufacturing sector, there is already an established nomenclature for the hierarchical levels, and this is reflected in RAMI4.0. In construction, no similar standard structure exists, and we have therefore chosen to instead use a generic and recursive structure that makes it possible to create different specific structures depending on the size and characteristics of the concrete processes to be carried out.

5.3 | Asset integration view

The asset integration view shows how a physical or other asset can be given a digital representation by providing it with an AAS. The AAS is thus a standardized way of adapting an asset so that it can become a CS in the SoS.

5.3.1 | Model

The main perspective provided by this view is where the AAS functionality should be allocated, depending on the characteristics of the asset. Certain physical assets, such as most machines used in construction, already contain embedded systems, often with a telematics capability that allows communication with other systems. In those cases, it is an option to place at least some of the AAS functionality in the embedded system. Other physical assets, such as piles of material, do not contain any embedded systems, and that is also the case for process assets. In those cases, the AAS needs to be placed on remote servers.

We have identified the following four typical cases, which are illustrated as UML deployment diagrams in Figure 5:

1. *Disconnected*. This is a situation where there is no connection between the asset and its AAS, such as when the asset is a pile of material. In this case, the AAS can contain information about the asset's status, but it needs to get the necessary data from other AAS, such as those of the machines that work on the pile of

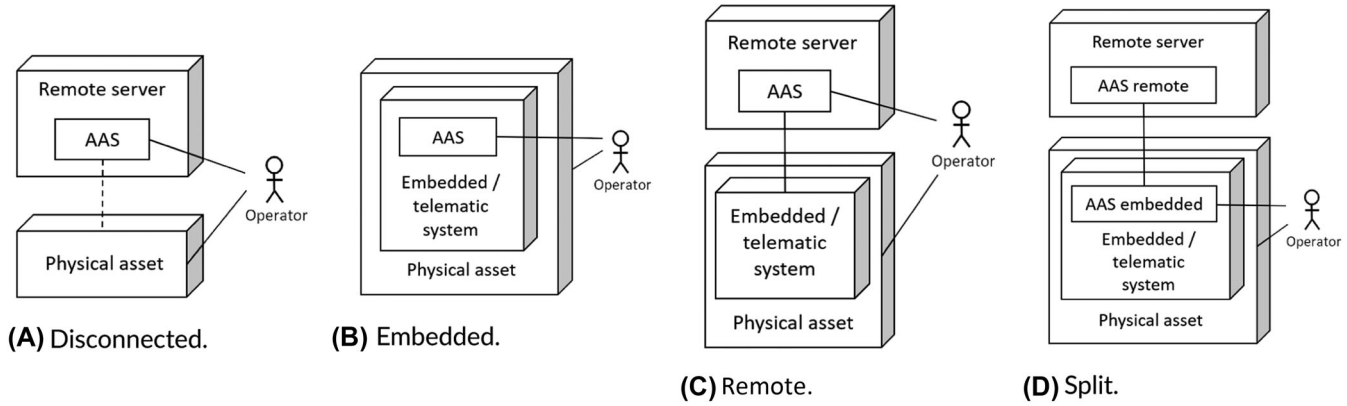


FIGURE 5 Asset integration view depicted as UML models

material, or from a human operator. This pattern also applies for a process asset, which does not have any physical representation. A third usage for this is when integrating machines from vendors that have not prepared them for usage within the SoS. Then, a user interface of the AAS can be used to communicate with the human operator, eg, through a handheld terminal, and the operator then monitors and controls the physical machine.

2. *Embedded*. When the physical asset contains an embedded system, the AAS can be incorporated as a software module in that embedded system, and interact with the physical asset directly through sensors and actuators. The communication with other AAS takes place through the telematics capabilities of the embedded system.
3. *Remote*. Another pattern is when the AAS is placed on a remote server, but it communicates with the existing embedded telematics system in the physical asset. This is useful if it is difficult to include more software in the embedded system. It can also be applied if the manufacturer of the physical asset does not want to open a direct interface to the machine for use by others for security reasons, and then the AAS can provide this interface on the server instead, and use a private link to the embedded system (eg, through protocols such as Open Platform Communication Unified Architecture that are frequently used in automation, or through proprietary protocols).
4. *Split*. The final pattern is a combination of the embedded and remote schemes, where the AAS is split into two parts, with certain elements located on the remote server and others in the embedded system. This is useful in situations where trade-offs are needed between performance, storage space, etc. Here, the operator would typically interact with the embedded part, whereas the server part is dealing with communication to other AAS and external systems.

5.3.2 | Concerns addressed

The asset integration view primarily deals with concerns related to the variability of assets, including the need to deal with manually, remotely, and autonomously operated machines, as well as the integration of machines from different vendors. It also relates to security of the embedded system.

5.3.3 | Rationale

The rationale for this view has to some extent been explained in connection to each pattern above. When it comes to remote operation, this is best handled by the remote integration, with the removal of the direct relation between the operator and the physical asset. Autonomous operation removes the operator completely, but instead, there is an interface from the higher hierarchical level that provides missions to the AAS.

5.4 | Communication view

The purpose of the communication view is to show what links exist for transferring information between AAS. Since we have adopted a recursive hierarchy, it covers all levels of hierarchy in RAMI4.0.

5.4.1 | Model

The architecture provides three types of communication, which are illustrated in Figure 6A through a UML class diagram. Here, the AAS class represents the shell connected to a specific asset, and the subclass composite AAS represents the shells around a process asset at a higher hierarchical level. The communication types are:

- *Publish-subscribe*. Each composite AAS, which has the role of coordinating a certain process, provides a communication infrastructure for all the participants of that process. The idea here is to use the publish-subscribe communication pattern,[21] where each CS of a process can publish messages. The other CS can choose to subscribe to messages of a certain kind, in which case they will be notified when new information arrives. The publish-subscribe server is the implementation of this communication infrastructure.
- *Direct*. A second communication option is to send messages directly between two AAS, and this is primarily intended to allow physical assets that are close to each other to exchange information over a short-range radio link (typically WLAN variants, ZigBee, or similar protocols).
- *External*. The third type of communication is with external users, which can be either operators of the asset or external systems. Here,

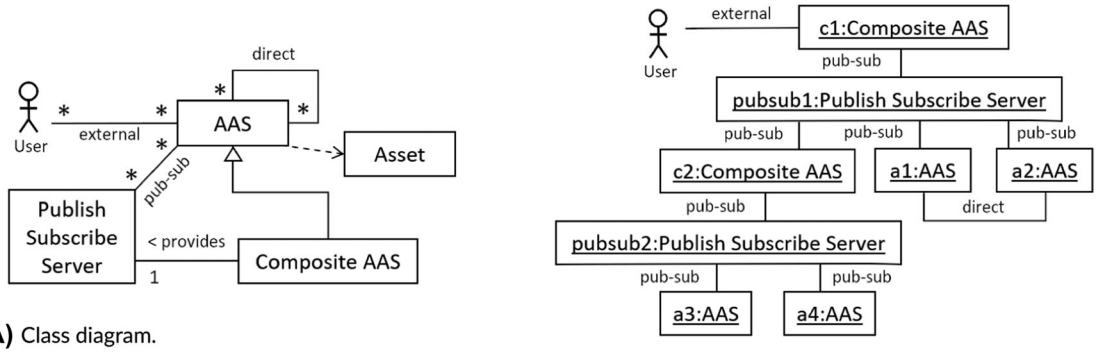


FIGURE 6 Communication view depicted as UML models

a client-server communication is intended, such as HTTP, or direct manipulation through a user interface.

Figure 6B contains a UML object diagram of an example situation. It shows how the composite AAS c1 provides a publish-subscribe server pubsub1 that allows communication with its subordinate AAS c2, a1, and a2. c2 is itself a composite AAS, and thus provides another publish-subscribe server pubsub2, to which the AAS a3 and a4 are connected. The direct communication is exemplified by the link between a1 and a2, and the external communication is shown between a user and c1.

5.4.2 | Concerns addressed

This view addresses the same functional concerns as the hierarchical view (see Section 5.2) when it comes to communication up and down the hierarchy, but here, the emphasis is on the mechanisms of the AAS, whereas the previous discussion focused on the logical relations. This view also covers the need for communication between assets that collaborate in a constellation, and the need for external users to communicate with assets, including the physical asset owner.

It also relates to several of the nonfunctional concerns, namely, security, since the chosen protocols come with specific security mechanisms; and flexibility and robustness, since the solution provides various means of communication that suit a range of situations. It also deals with the stability concern by relying primarily on existing and well-established communication standards, in particular from the web domain.

5.4.3 | Rationale

The motivation for using publish-subscribe communication within a process is that the CS involved typically collaborate closely within a certain physical area. They usually need to exchange information between all of them, and a point-to-point communication would then require the configuration of a large number of links. There is thus normally not a need to discriminate between them, and if there is such a need, it can be handled by dividing a process into subprocesses. The solution also makes it easier from a security management perspective, since a single credential allows communication with everyone in the same process.

Publish-subscribe communication has a number of benefits, and the asynchronous nature provides a robustness where middleware solutions can handle buffering in case of sporadic interruptions of communication. However, it does require cellular communication to reach a remote server, and sometimes, poor cellular communication coverage at construction sites poses a problem. This motivates the addition of direct communication between AAS, to allow collaboration between two closely positioned assets. It should thus primarily be seen as a backup solution, but could also be motivated for performance reasons in automated solutions where the delay of going through cellular communication and remote servers could be prohibitive. In this way, the appropriate operational trade-offs can be exploited by CS depending on the circumstances.

The client-server communication is mainly intended for user interfaces and external services, allowing different users to access information of an AAS asynchronously.

5.5 | Information view

The information view provides a core ontology defining the concepts used in the construction SoS. The information is represented as RDF, as described in Section 2.3, a format that can be used both for internal storage and communication.

Since many of the elements contained in the core ontology also have a corresponding element in the architecture, it is essential to notice the difference. As an example, an asset in the architecture is a physical item that exists in the real world. In the ontology, the RDF resource "Asset" is the name of the class of all assets, and a particular physical asset is given an individual RDF resource identifier by which it can be referred to. Such an RDF resource identifier should then be declared to be of the type "Asset."

5.5.1 | Model

The core ontology contains a small set of abstract concepts, represented as classes and properties. Most of these will in practice be used by subclassing them with more specific classes and properties. An overview of the concepts and their relations is given in the UML class diagram in Figure 7. The main concepts are:

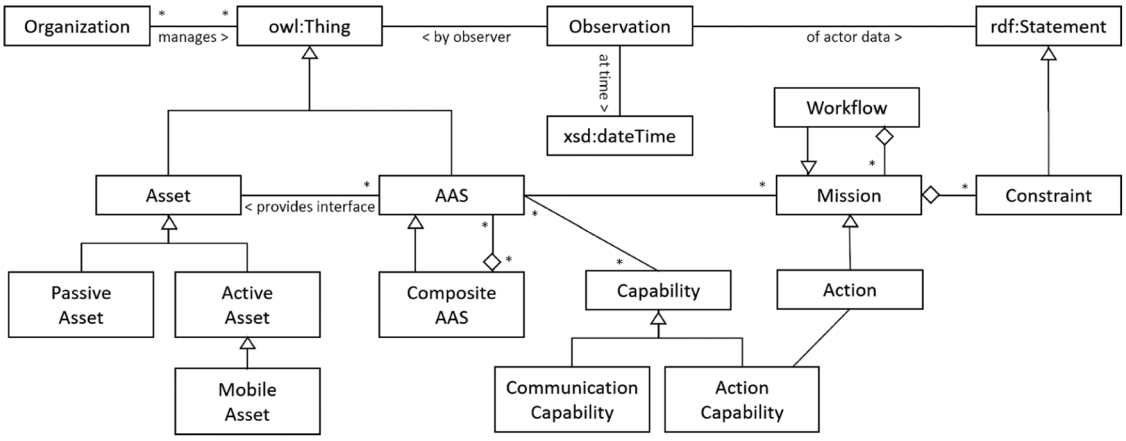


FIGURE 7 Information view showing ontology concepts depicted as a UML class diagram

1. *Asset.* The asset class is the type of the RDF resources representing asset instances. In our context, we make a difference between active and passive assets. *Active assets* can perform work, and are exemplified with different kinds of machines, as well as an overall site such as a quarry or road works site. A common subclass of active asset is a *mobile asset*, representing a machine that can move around. The active assets are thus the potential CS of the SoS. *Passive assets* cannot perform work, and represent items that are worked upon by the active assets, but which still have a value that needs to be represented. In the case study, examples of passive assets are the piles of material in the quarry, or the ground itself being worked on at the road site.
2. *AAS.* The AAS provides an information interface to the asset, and can be seen as a digital representation of an asset. Note that since it is possible to have several AAS for the same asset, it becomes important to view the asset and the AAS as separate RDF resources in the information model.
3. *Organization.* An organization represents a firm, or a part of the firm, and is useful to explain the managerial relations in the SoS. Examples of managerial relations are manufacturers, owners, or operators of assets.
4. *Capability.* A capability represents a function which the AAS can provide. This may refer to an *action capability* that is made available to the SoS through the AAS, such as the capability to move, for a mobile asset. It may also refer to a *communication capability*, which describes the addresses and protocols to use to exchange information with an AAS.
5. *Observation.* An observation represents a property value of an RDF resource that was observed or estimated at a certain time by a particular observer. It is thus a triple extended with the observer and the time, where a triple may be seen as a resource of type rdf:Statement. Observations can be used to collect data over time for later analysis, but can also have a value during operation. For instance, sometimes, an observation is transferred between AAS in several steps, and then it can be important to know where the data originate to assess credibility. Also, it can be important to know how old the observation is, to assess uncertainty.
6. *Mission.* A mission represents something that should be achieved. A mission may be an *action* or a *workflow* that contains other missions. Actions may not be broken down at this level of abstraction, and they correspond to capabilities of AAS. Missions may also have constraints.

As an example, a mission could be that an asset should move (an action corresponding to a capability of a mobile asset) to a specific geographic region and then drop its load there (another action corresponding to a capability of a load carrier.) This should be completed no later than a certain time (a constraint.)

A mission has a tree structure where the leaves are the atomic actions and workflows are the composite nodes. However, the workflows will also need to contain other kinds of information, such as expressing conditions, sequences, and parallel execution. These can conceptually be seen as constraints. In practice, a more user-friendly notation than RDF is used for creating missions, such as UML activity diagrams or Business Process Model and Notation (BPMN) diagrams.

When a composite AAS, which coordinates an SoS constellation, is given a mission whose actions correspond to its own capabilities, that mission usually needs to be decomposed, which means that the composite capabilities are translated into CS capabilities. The composite AAS thus needs to derive a new workflow that orchestrates the execution of CS. The parts of this new workflow that are to be conducted by a particular CS are then sent to it as a new mission.

5.5.2 | Concerns addressed

The information representation is mainly motivated by the concern for stability by providing a stable core vocabulary and by relying primarily on existing and well-established communication standards, in particular from the web domain. It also deals with flexibility, as well as the functional needs of representing missions, capabilities, and operational data.

5.5.3 | Rationale

The choice of RDF as a framework for information representation was guided by its expressiveness, which is needed to deal with the kind of

information processed as part of a construction SoS. For this expressiveness, RDF is the most widely used standardized notation, and thus provides stability, while at the same time being extensible enough to allow the needed flexibility. The same framework is also suggested for use in I4.0.[19]

5.6 | AAS composition view

The AAS composition view presents the structure of the AAS, and is related to the functional layer of RAMI4.0. However, RAMI4.0[15] does not provide sufficient details, but the structure has been outlined by Bedenbender et al.[18]

5.6.1 | Model

The purpose of the AAS structure (see Figure 8) is to make it modular, in order to support various kinds of communication protocols, capabilities, etc. This is achieved by decomposing the AAS into a number of submodels, which provide the specific implementations and interact with the core AAS through a generic Application Programming Interface. The following abstract submodel types are included, and are subclassed by concrete implementations:

- *Communication*. These submodels contain a common interface toward the AAS, and then translates calls to that interface to protocol specific versions. The different types of protocols, that each require a specific submodel, are mentioned in Section 5.4.
- *Capability*. Each type of asset has a specific number of things they are able to do as described in the ontology. For example, a hauler is able to transport a load, and offload it, whereas an excavator can pick up material, transport it, and offload it. Both can thus transport material, but the speed at which they can do it and the weight they can carry are vastly different. A compactor, on the other hand, cannot transport anything, but it can compress the surface it is moving over. The capability submodels provide an interface for performing a specific action.
- *Mission*. The capabilities are atomic in the sense that they cannot be broken down at this level of abstraction. Often, however, an asset needs to perform a combination of activities, which can be sequential, parallel, conditional, etc. To execute such a mission expressed in a certain formalism, a workflow execution engine is needed, and this is captured in the mission submodel. The reason for having this as a submodel, rather than a core part of the AAS, is that there could be different alternative formalisms for expressing missions, and they would then need different execution engines.

Apart from the submodels, the AAS also contains a world model, which is a database that can store RDF data, ie, triples containing subject, predicate, and object resource identifiers. The content of this was described in Section 5.5.

5.6.2 | Concerns addressed

The main concern addressed in this view is flexibility, which is achieved by having the extensible RDF formalism as a basis for the world model, and by allowing submodels to be added to an AAS. This modularization is also contributing to the stability of the architecture.

5.6.3 | Rationale

In our version of AAS, we have incorporated the ideas of submodels and ontologies from I4.0, since they contribute to flexibility, but extended it with the world model to emphasize data storage.

6 | DISCUSSION

After having presented the concrete construction SoS architecture description, this section discusses in general how well I4.0 supports adaptivity and interoperability of CS, and usages for SoS outside manufacturing. Finally, some additional experiences with respect to SoS engineering are reported.

6.1 | Constituent system adaptivity and interoperability

The mechanisms offered by RAMI4.0 when it comes to CS adaptivity are mainly related to the AAS submodel concept, which makes it relatively easy to extend the AAS. The standard is not totally clear on how flexible this mechanism is, but it appears to primarily provide flexibility during the design of the AAS to support product-line practices in the development of control systems, and is not intended for dynamic updates of new software.

To further increase adaptivity, the AAS submodel concept could be combined with a mechanism for installing software extensions, eg, by using an embedded Java virtual machine that allows the execution of dynamic software components within a real-time control system.[24] If the AAS is, at least in part, allocated in the cloud as suggested in Figures 5C and 5D, such extensions are even more feasible.

The main interoperability concepts in RAMI4.0 are the communication submodels, which make it possible to adapt to different protocols, and the use of ontologies and linked data to provide a common vocabulary. The latter is a very powerful concept, in particular

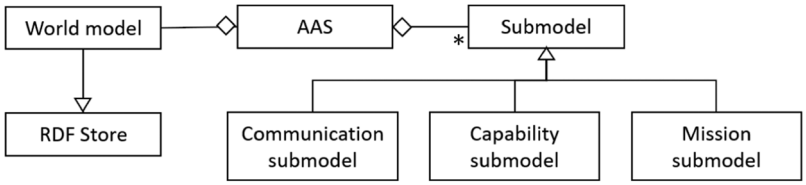


FIGURE 8 AAS composition view depicted as a UML class diagram

since the ontologies can be extended by individual CS and still, to some extent, be understood by others. However, it requires the establishment of a core ontology that is specific to the context of a particular SoS.

Traditional manufacturing systems are not SoS, and it is only with I4.0 that the SoS model is introduced. The manufacturing industry is thereby moving from a distributed hierarchical control system to a directed SoS, where there is still a fairly tight integration between adjacent levels in the hierarchy. In other SoS, such as the construction application studied in this paper, the connections are looser, and frequently a polyhierarchy appears where some CS are simultaneously part of several SoS. To deal with this, we have introduced the concept of missions and workflow submodels, which are not present in I4.0. This allows CS to operate more independently for a period of time, making their own operational decisions without needing to be in contact with the superior level in each step. The CS can then accept parallel missions from different superior SoS, and optimize the execution to find good trade-offs between conflicting missions. This has the further benefit that it gives a certain robustness in situations where communication is unreliable, because the CS can then continue with their missions autonomously while temporarily disconnected.

6.2 | SoS outside manufacturing

RAMI4.0 introduces useful concepts that could be applied in SoS from other industry sectors. However, it is important to be aware of some limitations. One is the directed nature of the manufacturing SoS already mentioned in the previous subsection, and alterations may be needed for acknowledged, collaborative, or virtual SoS.

Another difference is the life cycle, and this became apparent in the construction SoS. In manufacturing, a product is first designed, and then put into mass production, where the aim is to repeatedly produce as identical copies as possible time after time that gives a fairly stable process with limited evolution. The manufacturing consists of a sequence of work steps, where the material being worked on moves from one machine to another.

In construction, there is a design phase of, eg, a road, but the production phase consists of making just one copy of this road. During that operation, the material being worked on is the ground, and it is the machines that move to different places on the material rather than the other way around. The process is stable only for short periods of time and then evolves into something different.

6.3 | SoS engineering considerations

We will now discuss various subjective experiences that resulted from this work, and that relate mainly to SoS engineering methods and techniques.

To create an architecture description, a number of choices had to be made regarding the structure of that description. A major issue related to the selection of viewpoints, and ideally an existing architecture description framework would have been used. However, applying any of the common generic architecture frameworks such as Unified Architecture Framework or its predecessor leads to a lot of complexity and a difficulty of communicating it with nonexperts, and since only a small portion of such a framework would have been used, this idea was abandoned. We also looked extensively for specialized SoS architecture frameworks, but few could be found in the literature. In the end, we therefore decided to come up with our own set of viewpoints and connect them to RAMI4.0.

The next issue was which model kinds to include in the viewpoints, and here the options were primarily UML, SysML, or a SoS-specific notation such as SoS Architecture Description Language.[25] The main reason for choosing UML was that this work was carried out in close relationship with industry, where the participants were more familiar with UML than the other notations. UML also has a benefit in being more closely aligned with generic ontological concepts than the others, which was an important part in this work. Although the mapping from UML (such as Figure 7) to RDF was done manually, it was straightforward.

Throughout this project, we have worked in parallel on creating the architecture description and on implementing a prototype consisting of a simulation written in the Python programming language. This has been invaluable as a tool to experiment with different solutions, and for validating the consistency of the architecture description. The abstract I4.0 specifications leave much room for interpretation, and by making the interpretation concrete through an implementation, many aspects were better understood. The simulation was also useful for demonstrating to stakeholders some of the concepts in a dynamic way that is hard to achieve only with documents.

In the work, there has been a constant interplay between the SoS level and CS level. Finding the right trade-off between optimizing the SoS functionality and respecting the objectives and legacy of existing CS is essential, and the SoS design decisions that affect the CS architecture are a key here. The I4.0 AAS that wraps existing assets is a very useful concept in this, but may need to be complemented also with dynamic update mechanisms to enable rapid evolution of the SoS. The choice of semantic web techniques for information representation has also proven to be a very flexible solution for interoperability.

7 | RELATED WORK

In this section, some related work is introduced. Due to the multifaceted problem studied in the paper, a broad number of topics are relevant, and space only permits the inclusion of a few representative references on each topic.

7.1 | SoS architecture

Klein and van Vliet present a literature review on SoS architecture, surveying 200 papers in the field.[26] The paper focuses on a mapping of application areas, quality attributes, as well as impact of the research, but it does not provide any specific conclusions about applicable

patterns. It is worth noting that interoperability, security, and evolution are the three most prominent quality attributes identified in their study, which is also reflected in our work.

Selberg and Austin discuss architectural support for evolution of SoS, and claim that the change of a constituent should not affect the complexity of neither the SoS framework nor the CS.[27] They postulate that this requires standard interfaces, interface layers, and a continual verification and validation process. The first two are also present in our solution for the construction domain in the form of linked data as an information representation, and the publish-subscribe paradigm for communication.

Mori et al propose a SysML profile that contains a number of viewpoints each relating to a particular SoS concern.[28] The viewpoints include structure, time, dependability, security, evolution, dynamicity, multicriticality, and emergence. This provides an alternative structure to the one we have chosen, and would surely be valuable if the elicited concerns of an application matched those assumed by the proposed framework closer than it did in our case.

Axelsson and Kobetski discuss an architectural approach called federated embedded systems that provides adaptivity through a software plug-in mechanism.[24,29] In this way, the functionality of the CS may be dynamically adapted to the requirements of an SoS, and also to its evolution.

An SoS architecture aiming at the manufacturing domain is Arrowhead.[30] It takes a service-oriented approach, and addresses interoperability by providing adapters, which can either be allocated in a CS our outside it, in a similar way as we propose in the asset integration view. It applies publish-subscribe communication for event handling[31] and also has mechanisms for orchestration and service registries.

7.2 | I4.0 principles and applications

Hermann et al pinpoint the lack of common understanding and agreement on the principles behind I4.0, and therefore conduct a literature review of 130 papers on the subject.[6] Based on this, they claim that there are four principles that stand out, namely, interconnection, information transparency, decentralized decisions, and technical assistance for human operators. These principles also apply to our use of I4.0 in the construction domain.

A survey of I4.0 technologies and applications is provided by Lu, concluding that the concept is mainly used for smart factory and manufacturing, smart products, and to some extent also smart cities.[32] The current implementation status of I4.0 is discussed by Xu et al, and there appears to be a contrast between the advanced concepts proposed in the framework and what is actually used in practice, and they point at the need for more research on systems engineering to deal with the complexity of I4.0 systems.[33]

Logistics is a key issue in construction, but also in manufacturing, and the implications of I4.0 on manufacturing logistics have been studied by Strandhagen et al.[34] Their multicase study indicates that a high repetitiveness of production is essential for the applicability of I4.0 in this area. Hofmann and Rüsch also discuss logistics, but outside the manufacturing domain, demonstrating a potential for just-in-time delivery and cross-company Kanban systems, which is also relevant in construction logistics.[35]

Our paper contributes with respect to I4.0 mainly by investigating if the proposed concepts are general enough to support domains such as construction, which has different characteristics than manufacturing, and by studying their value as a foundation for SoS architecture in general.

7.3 | SoS and I4.0 in construction

Woodhead et al study the digital transformation of construction and argue that IT systems should not be considered as point solutions, but instead a business ecosystem approach is recommended based on Internet of Things and I4.0.[36] This provides a motivation for our research, and their paper identifies some technologies that are also used by us.

Oesterreich and Teuteberg survey the scientific and gray literature on I4.0-related technologies in construction, and show that building information modeling (BIM) is considered as a key enabler.[37] They also identify a number of challenges, among which are found knowledge management to deal with the fragmented characteristics of the construction value chain, and the lack of reference architectures. Both these aspects are dealt with in our paper.

Dallasega et al review the literature related to I4.0 for construction supply chains.[38] They confirm our view that construction logistics is to a large extent a temporary organization, which is a key difference to the more static manufacturing processes, and that this requires extensive coordination. They also discuss how I4.0 can improve the technological, organizational, geographical, and cognitive proximity in the supply chain.

In two related papers, it is investigated how ready the construction industry is for I4.0, both from a technological and a management perspective.[39,40] They highlight the use of collaboration and synchronization systems, but also point at a certain immaturity of this industry domain that requires partnering with the information and communication technology industry.

The application of SoS in construction has been studied by Zhu,[41] focusing on how to improve resilience in a construction project. The emphasis seems, however, to be on the planning phase, rather than the actual construction, whereas a related paper focuses on performance assessments at an abstract level.[42]

Bulbul et al discuss the application of SoS to intelligent construction, ie, the synergetic application of IT and communication to the construction project delivery processes.[43] Five subsystems are highlighted: the physical building, the virtual prototype, sensing systems, environmental systems, and human systems. The challenges identified include the decomposition of the constructed facility, interface design, and effective integration of technological and human subsystems. Matar et al study the three interacting SoS of the environment, the built facility, and the construction system, with respect to improving sustainability.[44]

7.4 | Interoperability based on ontologies and linked data

Abdalla et al present a systematic literature review of 31 papers on knowledge representation in SoS.[45] They observe that most papers describe ontologies for a particular domain, rather than for SoS in general. The motivations for knowledge representation are mainly to standardize terminology, ease integration, perform engineering activities, and support SoS management.

Baek et al present the M2SoS metamodel for SoS, which was elicited in the context of a mass causality incident response system.[46] The core ontology in our paper includes a subset of the concepts in M2SoS, but our work differs in its focus on a representation suitable for machine processing rather than for human readers.

Yahia et al discuss principles for how ontologies and description logic can be used to achieve interoperability in SoS.[47] They introduce the need for an "upper ontology," which is a role played by the core ontology in our approach. With this as a basis, two ontologies can be aligned through decision logic reasoning. The approach has been tested on two alternative ontologies from the manufacturing domain.

Curry proposes the use of linked data for dealing with SoS interoperability, using a dataspace architecture that allows for more heterogeneous and distributed storage and querying.[48] This is illustrated with an example from enterprise energy management. The approach is similar to ours, except that we are less focused on large-scale data management.

8 | CONCLUSIONS

In this paper, we have investigated through a case study in the construction domain the interplay between SoS and CS architecture. This includes both the design of an SoS architecture that takes into account the architecture of an existing CS, and the architecting of systems so that they can later become constituents of an SoS. These are important subjects as SoS become more and more common, and hence it can be expected that an increasing number of systems will at some point of their lifetime be modified to fit a certain SoS context. Particular aspects related to this are the adaptivity and interoperability of the systems, and this is an area where I4.0 is proposing certain solutions. We therefore investigated the suitability of I4.0 as a framework for SoS engineering in general, through a case study of a construction SoS.

We will now briefly summarize our findings with respect to the main themes in this paper:

- *Support of I4.0 concepts to the adaptivity and interoperability of CS.* A key idea of RAMI4.0 is to wrap the CS using an AAS, which provides a digital representation of a physical asset. The structure of the AAS consists of a number of submodels for communication and capabilities, which provide adaptivity. Through an ontology, a common vocabulary can also be provided to enhance interoperability. To these concepts, we suggest an extension to provide missions and workflow execution submodels that allow a larger autonomy of the CS, and we also make data explicit through a world model concept.
- *Support of I4.0 concepts to SoS outside the manufacturing domain.* To a large extent, we have found the I4.0 concepts useful in our case study, but there are also some areas where alterations are needed. This relates to less directed SoS than those from manufacturing, that may need a larger autonomy of the CS. Also, many SoS may have other evolution patterns than those of manufacturing, and involve far less stable constellations.
- *SoS architecture to support efficiency improvement in the construction domain.* In the paper, we have elaborated an architecture description based on ISO42010, which consisted of views related to hierarchy, asset integration, communication, information, and AAS composition. The foundation for this was RAMI4.0, but with suitable extensions and modifications based on the nature of the application. Many of the results are quite generic and the architecture description is likely to be relevant as a starting point for other hierarchically centralized SoS.

The validity of general conclusions based on a single case study is always debatable. Still, it gives some insights into generic SoS questions and also provides a solution for a concrete problem. This gives a more credible and in-depth understanding of some issues than does a superficial theoretical study, due to the complexity of reality.

There are numerous directions in which this research could continue. One of the most important is to examine the topic through other SoS cases from different application domains. Based on this, a number of stable patterns for CS architecture could be derived that can contribute to standards that in the long term greatly improve the possibility to quickly construct and evolve future SoS.

Regarding the case study, development continues with the objective of performing more detailed implementation leading up to demonstrations and validation in real production of the efficiency improvement potential. As part of this, the existing simulation is being refined with increasingly more detailed models, and eventually, the hierarchical communication structures will be transformed into an operations planning and management system for construction processes.

There is also a need for more work on how to describe the architecture of SoS and CS. Currently, there is a lack of architecture frameworks that are suitable for this. One potential route forward is to generalize the set of views described in this paper into viewpoints, which would constitute such a framework.

ORCID

Jakob Axelsson https://orcid.org/0000-0002-3986-1196
Joakim Fröberg https://orcid.org/0000-0001-8891-033X

REFERENCES

1. Maier MW. Architecting principles for systems-of-systems. *INCOSE Int Symp*. 1996;6(1):565-573. Available from: https://doi.org.wiley.com/10.1002/j.2334-5837.1996.tb02054.x
2. Dahmann J. System of systems pain points. *INCOSE Int Symp*. 2014;24(1):108-121. Available from: https://doi.org.wiley.com/10.1002/j.2334-5837.2014.tb03138.x

3. ISO/IEC/IEEE. Draft standard 21839 - systems and software engineering - system of systems considerations in life cycle stages of a system. ISO/IEC/IEEE; 2018.
4. ISO/IEC/IEEE. Draft standard 21840 - systems and software engineering - guidelines for the utilization of ISO/IEC/IEEE 15288 in the context of system of systems engineering. ISO/IEC/IEEE; 2018.
5. Schwab K. *The Fourth Industrial Revolution*. Geneva: World Economic Forum; 2016.
6. Hermann M, Pentek T, Otto B. Design principles for Industrie 4.0 scenarios. In: *49th Hawaii International Conference on System Sciences (HICSS)*. IEEE; 2016:3928-3937. Available from: https://ieeexplore.ieee.org/document/7427673/
7. Drath R, Horch A. Industrie 4.0: hit or hype? *IEEE Ind Electron Mag*. 2014;8(2):56-58. Available from: https://ieeexplore.ieee.org/document/6839101/
8. McKinsey & Co. Reinventing Construction: A Route to Higher Productivity. McKinsey & Company; 2017.
9. Axelsson J, Fröberg J, Eriksson P. Towards a system-of-systems for improved road construction efficiency using lean and Industry 4.0. In: *13th Annual Conference on System of Systems Engineering (SoSE)*. IEEE; 2018:576-582. Available from: https://ieeexplore.ieee.org/document/8428698/
10. Axelsson J. Experiences of using linked data and ontologies for operational data sharing in systems-of-systems. In: *IEEE Systems Conference*. Orlando, FL; 2019:396-403.
11. Hevner AR, March ST, Park J, Ram S. Design science in information systems research. *MIS Q*. 2004;28(1):75-105. Available from: https://www.springerlink.com/index/pdf/10.1007/s11576-006-0028-8
12. Axelsson J. Unified modeling of real-time control systems and their physical environments using UML. In: *Proceedings of the 8th Annual IEEE International Conference and Workshop on the Engineering of Computer Based Systems*. Washington, DC: IEEE Computer Society; 2001:18-25. Available from: https://ieeexplore.ieee.org/document/922399/
13. Axelsson J. Model based systems engineering using a continuous-time extension of the Unified Modeling Language (UML). *Syst Eng*. 2002;5(3):165-179. Available from: https://doi.org.wiley.com/10.1002/sys.10021
14. ISO. ISO/IEC/IEEE 42010: Systems and software engineering – architecture description. ISO/IEC/IEEE; 2011.
15. DIN. *SPEC 91345: Reference Architecture Model Industrie 4.0 (RAMI4.0)*. Berlin, Germany: Deutsches Institut für Normung; 2016.
16. Lu Y, Morris K, Frechette S. *Current Standards Landscape for Smart Manufacturing Systems*. Gaithersburg, MD: National Institute of Standards and Technology; 2016. Available from: https://nvlpubs.nist.gov/nistpubs/ir/2016/NIST.IR.8107.pdf
17. Grangel-Gonzalez I, Baptista P, Halilaj L, et al. The industry 4.0 standards landscape from a semantic integration perspective. In: *22nd IEEE International Conference on Emerging Technologies and Factory Automation (ETFA)*. IEEE; 2017:1-8. Available from: https://ieeexplore.ieee.org/document/8247584/
18. Bedenbender H, Billmann M, Epple U, et al. *Examples of the Asset Administration Shell for Industrie 4.0 Components – Basic Part*. Frankfurt am Main, Germany: ZVEI - German Electrical and Electronic Manufacturers' Association; 2016.
19. Grangel-Gonzalez I, Halilaj L, Coskun G, Auer S, Collarana D, Hoffmeister M. Towards a semantic administrative shell for Industry 4.0 components. In: *IEEE 10th International Conference on Semantic Computing (ICSC)*. IEEE; 2016:230-237. Available from: https://ieeexplore.ieee.org/document/7439338/
20. Berners-Lee T, Hendler J, Lassila O. The semantic web. *Sci Am*. 2001;284(5):28-37. Available from: https://www.jstor.org/stable/26059207
21. Ingram C, Payne R, Fitzgerald J. Architectural modelling patterns for systems of systems. In: *INCOSE International Symposium*. John Wiley & Sons, Ltd; 2015:1177-1192. Available from: https://doi.org.wiley.com/10.1002/j.2334-5837.2015.00123.x
22. Axelsson J. A refined terminology on system-of-systems substructure and constituent system states. In: *IEEE Systems of Systems Conference*. Anchorage, AK; 2019:31-36. Available from: https://ieeexplore.ieee.org/document/8753846
23. Dahmann JS, Baldwin KJ. Understanding the current state of US defense systems of systems and the implications for systems engineering. In: *2nd Annual IEEE Systems Conference*. IEEE; 2008:1-7. Available from: https://ieeexplore.ieee.org/articleDetails.jsp?arnumber=4518994
24. Axelsson J, Kobetski A. On the conceptual design of a dynamic component model for reconfigurable AUTOSAR systems. *ACM SIGBED Rev*. 2013; 10(4): 45-48.
25. Oquendo F. Formally describing the software architecture of systems-of-systems with SosADL. In: *11th System of Systems Engineering Conference (SoSE)*. IEEE; 2016:1-6. Available from: https://ieeexplore.ieee.org/lpdocs/epic03/wrapper.htm?arnumber=7542926
26. Klein J, van Vliet H. A systematic review of system-of-systems architecture research. In: *Proceedings of the 9th International ACM Sigsoft Conference on Quality of Software Architectures*. New York, NY: ACM Press; 2013:13. Available from: https://dl.acm.org/citation.cfm?id=2465478.2465490
27. Selberg SA, Austin MA. Toward an evolutionary system of systems architecture. In: *INCOSE International Symposium*. John Wiley & Sons, Ltd; 2008:1065-1078. Available from: doi.wiley.com/10.1002/j.2334-5837.2008.tb00863.x
28. Mori M, Ceccarelli A, Lollini P, Frömel B, Brancati F, Bondavalli A. Systems-of-systems modeling using a comprehensive viewpoint-based SysML profile. *J Softw Evol Process*. 2018;30(3):e1878. Available from: https://doi.org.wiley.com/10.1002/smr.1878
29. Axelsson J, Kobetski A. Architectural concepts for federated embedded systems. In: *Proceedings of European Conference on Software Architecture Workshops*. Vienna, Austria: ACM Press; 2014:1-8. Available from: https://dl.acm.org/citation.cfm?id=2642803.2647716
30. Delsing J. *IoT Automation: Arrowhead Framework*. Boca Raton, FL: CRC Press; 2017.
31. Albano M, Ferreira LL, Sousa J. Extending publish/subscribe mechanisms to SOA applications. In: *IEEE World Conference on Factory Communication Systems*. Aveiro, Portugal: IEEE; 2016:1-4. Available from: https://ieeexplore.ieee.org/document/7496528/
32. Lu Y. Industry 4.0: a survey of technologies, applications and open research issues. *J Ind Inf Integr*. 2017;(6):1-10.
33. Xu LD, Xu EL, Li L. Industry 4.0: state of the art and future trends. *Int J Prod Res*. 2018; 56(8): 2941-2962.
34. Strandhagen JW, Alfnes E, Strandhagen JO, Vallandingham LR. The fit of Industry 4.0 applications in manufacturing logistics: a multiple case study. *Adv Manuf*. 2017; 5(4): 344-358.
35. Hofmann E, Rüsch M. Industry 4.0 and the current status as well as future prospects on logistics. *Comput Ind*. 2017; 89: 23-34.
36. Woodhead R, Stephenson P, Morrey D. Digital construction: from point solutions to IoT ecosystems. *Autom Constr*. 2018;93:35-46. Available from: https://shura.shu.ac.uk/21244/
37. Oesterreich TD, Teuteberg F. Understanding the implications of digitisation and automation in the context of Industry 4.0: a triangulation approach and elements of a research agenda for the construction industry. *Comput Ind*. 2016; 83: 121-139.
38. Dallasega P, Rauch E, Linder C. Industry 4.0 as an enabler of proximity for construction supply chains: a systematic literature review. *Comput Ind*. 2018; 99: 205-225.
39. Maskuriy R, Selamat A, Ali KN, Maresova P, Krejcar O. Industry 4.0 for the construction industry—how ready is the industry? *Appl Sci*. 2019;9(14): 1-26.
40. Maskuriy R, Selamat A, Maresova P, Krejcar O, David OO. Industry 4.0 for the construction industry: review of management perspective. *Economies*. 2019;7(68): 1-14.

41. Zhu J. *A Systems-of-Systems Framework for Assessment of Resilience in Complex Construction Projects* [PhD dissertation]. University Park, FL: Florida International University; 2016.
42. Zhu J, Mostafavi A. A systems-of-systems framework for performance assessment in complex construction projects. *J Org Technol Manage Constr*. 2014;6(3): 1083-1093.
43. Bulbul T, Anumba C, Messner J. A system of systems approach to intelligent construction systems. In: *ASCE International Workshop on Computing in Civil Engineering*. Reston, VA. American Society of Civil Engineers; 2009:22-32.
44. Matar MM, Georgy ME, Abou-Zeid AM. Developing a BIM-oriented data model to enable sustainable construction in practice. In: *Proceedings of the 8th European Conference on Product and Process Modelling*. Cork, Ireland; 2010:79-87.
45. Abdalla G, Damasceno CDN, Guessi M, Oquendo F, Nakagawa EY. A Systematic literature review on knowledge representation approaches for systems-of-systems. In: *IX Brazilian Symposium on Components, Architectures and Reuse Software*. Belo Horizonte, Brazil; 2015:70-79.
46. Baek Y, Song J, Shin Y, Park S, Bae D. A meta-model for representing system-of-systems ontologies. In: *IEEE/ACM 6th International Workshop on Software Engineering for Systems-of-Systems (SESoS)*. Gothenburg, Sweden; 2018:1-7.
47. Yahia E, Yang J, Aubry A, Panetto H. On the use of description logic for semantic interoperability of enterprise systems. In: Meersman R, Herrero P, Dillon T, eds. *On the Move to Meaningful Internet Systems: OTM 2009 Workshops*. Lecture Notes in Computer Science, Vol. 5872. Berlin, Heidelberg: Springer; 2009:205-215.
48. Curry E. System of systems information interoperability using a linked dataspace. In: *7th International Conference on System of Systems Engineering (SoSE)*. Genova; 2012:101-106.

AUTHOR BIOGRAPHIES

JAKOB AXELSSON received an MSc in computer science in 1993, followed by a PhD in computer systems in 1997, both from Linköping University, Sweden. He was with Volvo Technological Development in Göteborg, Sweden, from 1997 to 2001, and then moved to Volvo Car Corporation in the same city, where he was responsible for research and advanced engineering of electrical and electronic systems between 2001 and 2010, and also became a 6 Sigma Black Belt. In 2004, he became a part-time adjunct professor at Mälardalen University, Västerås, Sweden, and in 2011, a full professor at the same university. Since 2010, he has also been with the Swedish Institute of Computer Science (SICS) in Kista, Sweden, where he founded the Software and Systems Engineering Laboratory, and is currently a senior research leader in systems-of-systems. He is the author of over 100 research publications. His research interests are focused on system architecture for embedded and cyber-physical systems, and system-of-systems engineering. Prof. Axelsson is a member of INCOSE, and has served as the chairman of the Swedish chapter. He received best paper awards at the IEEE International Conference on Engineering of Computer-Based Systems in 2008; Euromicro Conference on Software Engineering and Advanced Applications in 2014; and IEEE International Symposium on Industrial Embedded Systems in 2017.

JOAKIM FRÖBERG is a senior researcher at RISE Research Institutes of Sweden. He earned his PhD in computer science from Mälardalen University in 2007 on the topic of engineering automotive electronic systems and he has over 20 years of experience in developing software-intensive embedded systems. Dr. Fröberg's research interests include engineering of complex computer-based systems, especially methods for analysis and selection of system architecture and systems-of-systems, and also process and requirements related to engineering and functional safety certification. He has participated in several automotive development projects including fuel-electric hybrid control systems and autonomous control systems for transport, automotive, mining, and agricultural applications.

PETER ERIKSSON holds a BSc in electrical engineering from Dalarna University, Sweden, in 1994. He joined Ericsson Radio Systems in Gävle, Sweden, during 1995-1999. There, he was responsible for system test solutions of the Base Station Controller (BSC)/Base Transceiver Station (BSC), and for measurement methodology and verification concepts in Ericsson's 3G demonstrator. In this position, he was also part of a research team in RF measurement technology. In 1999, he started at the Nokia R&D center in Kista with the task to form a new R&D team to develop measurement practices and a verification platform. Between 2001 and 2006, he was with ABB as a project manager for Process Automation Equipment, driving projects with Safety Integrity Level (SIL) 3 and 4. Between 2007 and 2008, he was responsible for driving organizational efficiency improvements at the company Hästens beds. In 2008, he joined Volvo Construction Equipment's Component Division in Eskilstuna, Sweden, driving the research and predevelopment strategy and portfolio. In 2009, he moved to a new position at Volvo Construction Equipment managing the global research portfolio and corporate long-term technology strategy. He was also part of a core team developing the Volvo Group Technology Plan. In 2019, he started as a Senior Advisor for Blue Institute, a think tank supporting organizations in understanding and forming their futures. He is a certified future strategist, ICFS, has a CMI Level 7 Award in Strategic Management and Leadership and is a certified foresight IFTF practitioner.

How to cite this article: Axelsson J, JFröberg, PEriksson. Architecting systems-of-systems and their constituents: A case study applying Industry 4.0 in the construction domain. *Systems Engineering*. 2019;22:455–470. https://doi.org/10.1002/sys.21516