Experiences of Using Linked Data and Ontologies for Operational Data Sharing in Systems-of-Systems

Jakob Axelsson
Mälardalen University and Research Institutes of Sweden (RISE)
Västerås, Sweden
jakob.axelsson@mdh.se

***Abstract*—This paper addresses the problem of exchanging complex data between the constituent systems in a system-of-systems. This is necessary to ensure that they have compatible understandings of the world surrounding them and entails a need for semantic interoperability between the constituents. Through a case study of a road construction system-of-systems, the world wide web technologies of linked data and ontologies are explored as a framework for data representation and exchange. This data includes several broad categories, such as assets, interfaces, organizations, capabilities, missions, and observations, as well as various properties of those. It is also discussed how the constituents can use this data for reasoning and decision making. The results have been validated through a simulation-based research prototype of the road construction case, from which experiences are reported.**

Keywords—system-of-systems, linked data, ontology, semantic interoperability, road construction, Industry 4.0.

I. Introduction

During the operation of a *system-of-systems* (SoS), the participating *constituent systems* (CS) try to collaborate in order to provide some value which they cannot create individually. Such a collaboration requires a certain sharing of *world views* between the CS's, i.e., the model that each CS has of the context it is operating in. This context includes the overall SoS, the other CS's, itself, and their environment. external to all of them.

Due to the *operational and managerial independence* of the CS's [1], some participants will have exclusive access to information which would be useful to others. For instance, each CS will exclusively know its own internal state, and the data gathered through its different inputs. Therefore, the CS's need to engage in a continuous exchange of data to let each of them construct and update a world view that covers what it needs to fulfil its role in the collaboration. Such an exchange requires interoperability on all levels of communication, and in particular, *semantic interoperability* is required to ensure that the different CS's understand a shared piece of information in a similar way.

The overall contribution of this paper is to investigate, through the application of a real-world SoS example, what is needed to achieve this alignment of world views across CS's during their operation. In particular, it studies the use of *linked data* and *ontologies* to provide interchangeable information representations within an SoS. These are concepts made popular through the *semantic web* initiative [2], which extends the syntactic information on world wide web (WWW) pages with semantic information. This allows systems to reason about the contents, and link different information sources together. The technologies used for this are universal, and not limited to the context of the web, but can be used also for knowledge representations in other contexts. A hypothesis of this work is that semantic web technologies are a suitable basis for SoS semantic interoperability in general due to its decentralized approach and wide acceptance.

This research was funded by 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.

A. Research Questions and Method

The paper studies the following main research questions:

1. What kind of information needs to be shared by CS's during their operation within an SoS?
2. Are linked data and ontologies effective representations of the required information?

This research is constructive, in the sense that the ultimate objective is an artifact in the form of an information representation concept. Therefore, a research method based on *Design Science* is suitable [3]. In this method, the desired artifacts are developed and evaluated in an iterative process, where the evaluation is based on both interacting with the relevant environment and with the existing knowledge base. The interactions with the environment are particularly important and have been handled through an industrial case study of an SoS for road construction. In that study, preliminary results have been presented and discussed with domain specialists, and a prototype software implementation simulating the SoS has also been used to validate that the concepts could work in practice. In addition, existing research literature and standards have extensively been used to guide the work.

B. Overview of Paper

The remainder of the paper is structured as follows: In the next section, the semantic web concepts of linked data and ontologies are described. Then, in Section III, the road construction SoS case study is outlined, followed in Section IV by an investigation of the kind of information that needs to be represented and shared in the world models of CS's in that SoS. Section V presents how the information is managed during operation and how it can be used for reasoning and decision making in the individual CS's. In Section VI, the results are discussed, and Section VII introduces some related research. In the final section, the conclusions are summarized together with indications of future extensions of the research.

II. Linked Data and Ontologies

As mentioned in the introduction, this work uses semantic web concepts as a foundation for SoS information representation. The main building pieces of this is a generic data representation called linked data; techniques for storing and retrieving data; a framework for building ontologies that contains general concepts for knowledge representation; and a number of more or less standard vocabularies. Each of these will be introduced in the following subsections.

A. Linked Data Representation

The Resource Description Framework (RDF) is the WWW "framework for expressing information about resources, where a resource can be anything, including documents, people, physical objects, and abstract concepts" [4].

Resources are represented by International Resource Identifiers (IRI), which are strings with a syntax very similar to the URL strings used for web addresses. The strength of this is that unique IRIs can be generated in a distributed way, by following a principle where the domain name of an IRI should be owned by the organization that creates the IRI.

The information about the resources is expressed as very simple statements on the format subject – predicate – object *triples*, where the predicate is also called a property. The subject and predicate are always resources, and the object may be either a resource or literal data (e.g. strings, number).

As a very simple example, assume that an organization needs to represent information about cars, and express that a particular car is red and is driven by a certain employee. The resources in question are the car and the driver, which we could assign the following IRIs:

- Car: <http://example.com/car/123>.
- Driver: <http://example.com/employee/456>.

To enhance readability, it is possible to introduce abbreviations for common IRI prefixes. For example, we could use "ex" to be an abbreviation of <http://example.com/>, and the car in the example can then be referred to as ex:car/123. This abbreviation can be seen as introducing a namespace for resources.

The following triples capture the required information:

- ex:car/123 ex:has_color "red" (where the object is a literal).
- ex:employee/456 ex:drives ex:car/123.

The set of triples is all that is needed to represent the information. More formally, a knowledge base K is a set $K \subseteq \{R \cup B\} \times R \times \{R \cup L \cup B\}$, where R is the set of IRIs, L is the set of literals, and B is the set of anonymous resources (also known as blank nodes). The blank nodes are useful when we do not want to express the identify of a resource but are only interested in what combination of properties it has. Since the same resource may appear in several triples in K, this set can be seen as a graph where the edges are the triples and the nodes are the (implicit) set of all IRIs, literals, and blank nodes that appear in any triple. It is common to refer to a set of triples as an (RDF) graph.

Note that since the representation is just a set of triples of resources with unique identifiers, basic set operations such as union can be applied, e.g. to merge two knowledge bases, which is quite convenient when communicating linked data between systems.

B. Database Queries and Serialization

To store larger amounts of RDF triples, databases are normally used, and these can be queried in different ways. The simplest way is pattern matching, by specifying a triple where one or several of the fields are replaced by a wildcard. The result is then the set of all triples that match this pattern. However, sometimes more advanced queries are needed that relate several resources to each other or put more complicated constraints on the relations. For this, the SPARQL query language has been developed [5].

To exchange data between systems or store data in files, serialization formats are needed, and a number of textual notations for this exist, such as RDF/XML [6] or more human-friendly notations such as Turtle [7] (which is also used in the examples in this paper).

C. Generic Ontological Concepts

The RDF linked data representation can be used to construct ontologies, i.e., definitions of concepts, categories, properties, and relations between the resources. In other words, it creates a terminology to be used, and this is needed to represent a common domain of knowledge. However, the definition of ontological concepts typically uses a small set of logical constructs, that have for this reason been standardized and are universally accepted. Those constructs come from logic, in particular description logic, and include:

- *Class and instances.* A resource may refer to a class of objects, and it can be expressed that another resource is an instance of the class (or that the class is the *type* of the instance).
- *Property:* A resource may be a property, indicating that it makes sense to use it as the predicate of a triple. For a property, it can be expressed that it has a certain *range* and *domain*, thereby restricting the subjects and objects to certain classes.
- *Subclasses and sub-properties.* It can be expressed that one class is a subclass of another, implying that an instance of the first subclass is also an instance of the more general class. Likewise, a property may be a sub-property of another property.

These concepts are introduced in the RDF Schema (RDFS) specification [8], which uses the namespaces "rdf:" and "rdfs:". With these concepts, we could in the car example above express that our car belongs to a class called Car, which is a subclass of vehicles, and that there is a property "has_color" whose domain is the class of cars:

- ex:car/123 rdf:type ex:Car.
- ex:Car rdf:type rdfs:Class.
- ex:Vehicle rdf:type rdfs:Class.
- ex:Car rdfs:subClassOf ex:Vehicle.
- ex:has_color rdf:type rdf:Property.
- ex:has_color rdfs:domain ex:Car.

Thus, if one system transfers some data about a car to another system, which is not aware of what a car is but knows the more general concept of a vehicle, the other system may still make use of the information about the instance given the ontological facts above stating that a car is a vehicle.

RDFS contains a small set of constructs which are very common. An elaborate extension is in the Web Ontology Language version 2 (OWL2, with the namespace "owl:") [9] that allows the expression of much richer information about resources, classes, properties, etc. It is beyond the scope of this short overview to go into any details, but it should be mentioned that the OWL2 constructs have well-defined meanings in terms of logic and can thus be used to logically infer more information from what has been explicitly provided.

It should be noted that all the meta-concepts introduced in RDFS and OWL2 are just ordinary resource IRIs and do not in any way change the linked data representation. They have a special place only in the sense that they define very common concepts that are universally agreed upon. Also, it should be noted that many RDFS concepts are similar to those used in UML class diagrams, and it is not uncommon to use UML to provide a visual overview of an ontology for human readers.

D. *Common Vocabularies*

Apart from the meta-concepts from RDFS and OWL2, certain other more concrete concepts tend to reoccur in many domains, and a number of recommended vocabularies have been developed. For example, the Dublin Core (DC) vocabulary contains concepts related to information resources [10] and the Friend of a Friend (FOAF) vocabulary defines terms related to people and their relations [11]. It is not mandatory to reuse such vocabularies, but it does make life simpler if at some point, data from one system is to be shared with another system, since it increases the chances that they use the same terminology for similar concepts.

III. Case Study: A Road Construction SoS

To investigate the use of linked data and ontologies for SoS semantic interoperability, we have conducted an industrial case study where the concepts have been tried. The case comes from the construction domain, where a number of different machines and other assets form the CS's that need to collaborate on a worksite to produce a road. The case has been described in more detail in a previous paper [12], and only a brief summary will be provided here to give the context.

A. *Purpose*

The construction sector is one of the largest industries in the world, with an annual turn-over of around 13% of the global GDP [13]. In this case study, we focus on road construction, which is a significant portion of this. However, whereas other industries such as manufacturing have seen 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 [13].

The production at a construction site involves a number of machines and other assets with complementary capabilities, that need to collaborate, and it thus constitutes an SoS. In our research, we make a hypothesis that the productivity gap 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 are investigating how machine-oriented communication and supporting tools can supplement the human communication, in order to improve coordination and identify wastes that should be eliminated.

B. *Architectural Concerns*

The construction SoS has several architectural concerns, including:

- *Multi-vendor*. Machines from different vendors and of different types must be able to collaborate on the site.
- *Autonomous and manual*. Current road construction equipment is mostly manually operated, but there is a strong trend to develop more automated solutions. The SoS architecture must handle a mix of both types.
- *Secure*. Participating in an SoS requires a certain degree of openness, and it must be assured that confidential information of a certain participant does not become accessible to others.
- *Flexible*. A difference between road construction and manufacturing is the continuous changes in the former. The process has much shorter periods of steady state, which makes process optimization more difficult. This increases the need for up-to-date information, support for re-planning and reconfiguration. The variability between different construction projects is substantial.
- *Robust*. 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.

C. *SoS Architecture*

Based on these concerns, a number of architectural principles have been identified, which are based on similar ideas as used in the Reference Architecture Model for Industry 4.0 (RAMI4.0) [14]:

- *Asset administration shell (AAS)*. In order to provide a common interface to constituent systems, RAMI 4.0 introduces the AAS concept, which can encapsulate an asset such as a physical machine and give it proper information interfaces. This allows for different assets to communicate in a standard way, and also provides mechanisms for self-description.
- *Hierarchy*. Construction work is today organized in a hierarchy, where the working machines are at the bottom. The next layer is the work site (or a part within it). Above that is a project level, which can coordinate several sites (e.g. a road site, a quarry, and an asphalt plant). However, this structure is in fact usually a poly-hierarchy, where certain parts can serve several parents simultaneously. The different parts in the hierarchy are usually ran by different organizations, resulting in an operational and managerial independence. In our proposed approach, elements on all levels are treated as assets, and given their own AAS to handle interactions.
- *Capabilities and submodels*. The different assets are described in terms of their capabilities, i.e. what services they can provide. For each capability, there is a sub-model that implements the service, making the design of the constituent systems modular. Capabilities include the ability to use different communication techniques, but also different physical work that can be done depending on the machine type.
- *World models*. Each constituent system of the SoS will contain a substantial amount of information about other constituents, as well as data about the environment they operate in. We call this information set the asset's world model, and at times it is essential

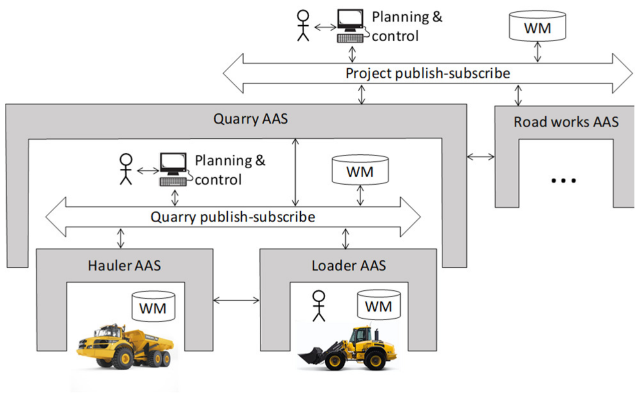

Fig. 1. Overview of the road construction SoS architecture.

to extract data from the world model and exchange it with other assets.

- *Publish-subscribe communication.* Within each sub-process in the hierarchy, there is usually a need for all involved parties to communicate with each other. We have solved this by using a publish-subscribe system that all the concerned AAS's connect to.

The key elements of the SoS architecture are illustrated in Fig. 1.

D. Validation Prototype

To test that the different architectural concepts work together and are effective, a research prototype has been built. It is implemented in the Python language and is distributed to allow different parts to be allocated to different computers. Communication is over HTTP and the publish-subscribe protocol AMQP [15], and these are implemented as submodels, as are the various capabilities of different machines. The world models are implemented using in-memory triple stores from the Python library RDFLIB.

The main conceptual difference compared to a real system is that the physical assets are simulated. This also includes the human operators of machines, which means that all simulated machines behave as if they were automated.

IV. SoS Ontology Content

We will now introduce a set of ontological concepts that are useful when describing an SoS. The concepts are generic, and should be relevant to most SoS, but as they are derived from the case study in the previous section, we cannot say for sure if they are universal or not. The ontology is divided into a core part, which represents generic concepts that must be agreed upon by all CS's in the SoS, and extensions, that allows a CS to specify more specific information that others may or may not use.

A. Core Ontology

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 Fig. 2.

1) Asset. In the terminology from Industry 4.0, an asset is any "object which has a value for the organization" [14]. A central concept of Industry 4.0 is that "assets can be combined in any way, and these assets are formally described in sufficient detail for use in the digital world". 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 e.g. a machine that can move around. The active assets are thus the potential CS's 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 asset administration shell provides an information interface to the asset, and can be seen as a virtual digital and active representation of an asset. The AAS is a standardized way of adapting an asset so that it can become a CS in the SoS (it represents the CSΔ, in the terminology of [16]). Note that it is possible to have several AAS's for the same asset, and it is thus important to view them as separate RDF resources in the world models. Also, an AAS may be *composite*, if it encapsulates a number of other AAS's. A composite AAS thus represents an SoS level in the hierarchy.

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 the manufacturers, owners, or operators of different assets.

4) Capability. A capability represents a function which the AAS can provide. This may refer to an *action capability* which 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's in several steps, and then it can be important to know where the data originates 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* which contains other missions. Actions may not be broken down at this level of abstraction, and they correspond to capabilities of AAS's. Missions may also have *constraints*.

As an example, a mission may 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.)

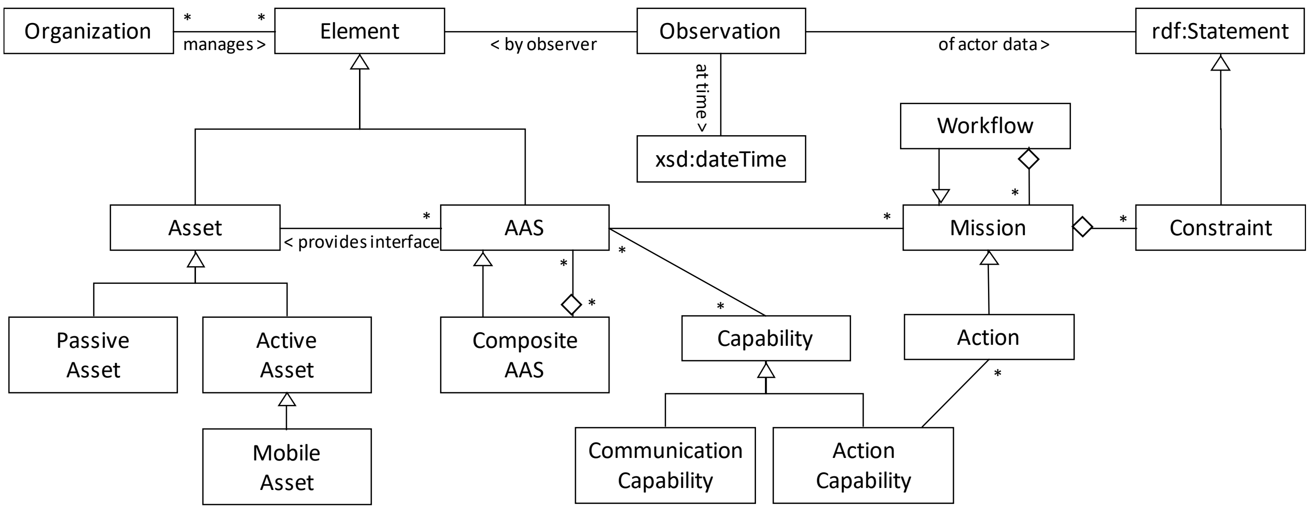

Fig. 2. UML class diagram illustrating the concepts in the core ontology.

We will not specify further how constraints are expressed but note that OWL2 contains concepts that can express e.g. bounds on numerical values, and since these are represented as triples, a constraint is a subclass of rdf:Statement.

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. However, in practice a user-friendly notation is needed. We have experimented with various alternatives, including graphical languages such as the Business Process Model and Notation (BPMN) [17], but we leave the choice of notation open in the ontology.

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

B. *World Models and Extensions*

The core ontology is used to structure the data in the world model of each AAS. This world model is thus a set of RDF triples, where the resources include those specified in the core ontology.

A specific AAS, that encapsulates a concrete asset, may extend the core ontology with new concepts. As an example, the AAS of a working machine such as an excavator would introduce an ontology class of the physical excavator, which is a subclass of a mobile asset. Further, it would define different properties that are relevant, both dynamic such as the current load it is carrying, and static such as the maximum load it can lift. Also, it would describe itself, i.e., what capabilities it provides, etc.

An important aspect of an AAS in Industry 4.0 is that it can provide a *manifest*, i.e., a description of itself. In our implementation, this description essentially corresponds to the extended ontology it uses, plus its current state. By requesting the manifest of an AAS, using one of its communication capabilities, a serialized set of RDF triples is transmitted, which can be added to the receiving AAS's world model. In this way, a common language that can be used for their communication is created.

V. LINKED DATA USAGE AND MANAGEMENT

Having defined the common and extended ontology, these can be put into use to let a CS build its world model and exchange information about it. This section provides more details on how the constituent systems create and use the linked data, as well as some experiences on data management.

A. *Operational Usage of Linked Data in Constituents*

The AAS primarily has two tasks: to monitor and control the asset, and to communicate with other AAS's. The monitoring part consists of gathering or estimating data over time, based on events or sampling. The data is stored in the world model as observations, according to the ontology. It is also published on the publish-subscribe bus of the composite AAS. It is up to each AAS to decide what information on this bus they want to subscribe to. When they choose to subscribe, they can also filter out what parts of the received RDF graph they want to include in their own world model.

The control part is more intricate, since it involves decision making. For most assets in the case study, there is a human operator that makes the decision, and the AAS therefore needs to present data and analyses of data to a human.

As an example, consider a quarry site, which has a composite AAS which contains the AAS's of machines on the site. The site AAS receives a mission to produce a certain quantity of aggregate of a specific granularity. It will deal with this by using a defined workflow if one exists or letting a site operator design a new workflow if necessary. This workflow contains actions where capabilities of different machines are needed, and the site manager must then allocate missions to suitable machines. To do so, the world model must be consulted and queried for which machines can perform a certain capability.

An issue is how to create user interfaces in a composite AAS, where e.g. the site manager wants to inspect properties of the included AAS's. However, those AAS's may have extended the ontology with new specific classes and properties, and the user interface has to be built dynamically. This can be done by querying the manifests to find out what properties (i.e., resources of type rdf:Property) exist in this particular AAS and use that to display data in an understandable way.

B. Process Improvement

The previous subsection discussed how to use the linked data during operation, but an important usage is also related to process improvement. The motivation for the construction SoS is to improve productivity, and a common approach for dealing with such problems is the Lean methodology. The main steps are to create a process description; collect operational data; analyze it to identify possible wastes (activities that do not create value); and update the process to remove wastes.

In our approach, there is a potential for automating a large part of this process, which is today usually done manually. This is achieved by collecting observations of asset and AAS states over time, such as where they are, what actions they are performing, and key operational properties such as the mode they are in. From this, waste patterns can be identified, such as excessive inventories; unnecessary machine movements; transportation; waiting; etc. These can be precisely attributed to workflow steps, since the current action is also observed, and to geographical locations.

It is very important to automate this analysis process since construction processes tend to be stable only for a short period, and in many cases would change before a manual process had the time to complete.

C. Data Management

A first step in using RDF is to create suitable IRIs for the asset and AAS instances. For tangible assets like machines, the most natural is to let the original equipment manufacturer (OEM) mint a suitable IRI. One way of doing this is to combine the domain name of the OEM with the serial number of the machine.

For more temporary passive assets, like a pile of rocks in a quarry, the site owner would need a process for finding suitable IRIs. For instance, they could combine their domain name with the position and creation time of the pile, to ensure uniqueness. For an AAS, a suitable IRI could be the IRI of the asset it refers to, appended with "/aas" or similar. For ontology extensions, such as defining new asset types or capabilities, the domain name of the organization that defines the extension is also suitable as a basis, appended with strings related to the name of the asset type or capability. In this way, the manifest of an AAS is easily generated.

It is practical to divide the world model into several graphs, representing different types of information. This is because some subsets of information may have a limited validity, and it is then easy to delete the entire graph related to that information. Also, some subsets are interesting to share with others, which is easier if they are already in a separate graph. It is common for linked data stores to be able to handle a larger graph being divided into subsets, and to allow searching for data either in the entire graph or in one of the subsets. Examples of data we have put in subgraphs is the base ontology; the extended ontology; the manifest of the AAS; the most recent observations about the asset's state; capabilities of the asset; state information about other assets; and tasks received from other AAS's.

VI. Discussion

The previous sections have presented concrete solutions that were derived as part of the implementation work for our research prototype. That work however also generated a number of questions, whose answers are not yet clear. In this section, we will discuss a few of those issues and provide some of our subjective experiences.

A. Suitability of Linked Data for Different Information

In general, we have found the RDF approach to linked data and ontologies easy to use and quite natural. However, there are some types of fine-grain data where it may not be the appropriate solution. This includes pixel-based images, for which semantic information about what they represent is not available. In the construction domain, geographical and geotechnical information is central, both for the surface (topology, routes, destinations) and underneath it (capturing the materials, such as rock, soil, etc.). As it seems, the industry has not yet agreed on an appropriate data storage format, but is mostly considering document-based storage, using e.g. the LandXML schema. Probably, hybrid solutions are needed where specific formats are annotated with semantic data.

B. Level of Abstraction

Many SoS are hierarchical in their nature, and it has proven to be challenging to find the right level of abstraction for each level. This has become obvious when it comes to mission and their decomposition into actions, which correspond to capabilities in the CS's. If the SoS level provides very detailed tasks to the CS's, it may be able to orchestrate their actions with a fine granularity and in that way optimize the overall SoS performance. However, by the first law of cybernetics [18], this requires access to a large amount of information about the CS's, which may be impractical. The opposite is to give the CS's much vaguer tasks, which allows them to optimize within those bounds, and make appropriate compromises vis-à-vis their own objectives. The risk is however that the emergent properties of their interactions are not as desired.

It would be interesting to explore this topic further and extend the ontology to allow CS's to express what their priorities are, that guide how they make decisions. Possibly, this could make it more transparent how a CS will deal with a more abstract task and make it possible to find an appropriate level of abstraction.

C. Reasoning

The foundation of notations such as OWL2 is decision logic, and this means that it is possible to reason logically about the content of a linked data set using tools such as [20]. In the prototype, we have used rudimentary ad hoc procedures for reasoning, to solve situations such as finding all property relations defined for a class, including those inherited from super-classes. This approach reduces the dependencies of the software to third-party modules, and can reduce the hardware resources needed, and it may suffice as long as only a subset of OWL2 is used. However, if certain extensions use more powerful expressions, there is a risk that relevant information is missed or misinterpreted.

One should however be aware that even the powerful OWL reasoners are not complete, and do not have the capability to deal with complex spatial or temporal data, which is inherent in our application domain.

D. Efficiency of Data Processing and Storage

Representing data using RDF may seem costly in terms of memory and communication bandwidth, due to the very verbose IRIs. However, we have not found this to be a problem, since the triple stores compact the representation, and the number of triples that are repeatedly communicated tends to be limited. Larger amounts of data primarily occur when collecting observations over time but given the fact that commercial triple stores can deal with billions or even trillions of triples [19], even this should not be an issue.

The architecture depicted in Fig. 1 can give the impression that the AAS and world model need to be implemented in the embedded systems of the assets, but this is not necessarily the case. A common solution is likely to be that the AAS is implemented in the cloud and has a private communication channel to the embedded system in the asset. This removes concerns about dealing with RDF in memory constraint systems.

An open question is related to how data is to be communicated. The publish-subscribe model in the architecture makes handling of the communication straightforward, but it does not specify whether the AAS's should repeatedly send updates of all parts of their asset state, or only send updates about things that change. Depending on the dynamics of the application, this could have large effects on bandwidth consumption, but more rare updates could imply that newly added AAS's do not get access to the complete state until information changes. A suitable compromise is probably a combination of requests for complete information when needed and continuous updates of changes.

When dealing with large data sets, such as maps or geotechnical information, it might be appropriate to only communicate those parts that are needed for a particular task. However, this raises questions how to extract that data from a larger database.

VII. Related Work

The ideas of using ontologies in the context of SoS is not new, and the existing body of research can broadly be divided into general work on the need of SoS ontologies; specific suggestions on concrete ontologies for different SoS aspects; and implementation issues in an operational context.

A. General Work on the Need of SoS Ontologies

Several authors have looked at ontologies for SoS, with the purpose of clarifying terminology and concepts. Langford et al. [21] investigate the philosophical underpinnings of ontologies for systems and SoS. Based on a set of axioms from set theory, the ontology is divided into an objective and a subjective part, that are then used to form an integrative framework. It is also noted that the temporal domain is central and can be used to understand emergence.

Abdalla et al. [22] present a systematic literature review of 31 papers on knowledge representation in SoS. 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, to ease integration, engineering activities, as well as SoS management.

Sarder et al. [23] discuss the processes of ontology development, and apply it to systems engineering, mainly with the motivation to clarify terminology. A similar need is also identified for SoS engineering, but without providing any concrete suggestions.

B. Concrete Ontologies for SoS

Baek et al. [24] present the M2SoS meta-model for SoS, which was elicited in the context of a mass causality incident response system. 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.

Bénaben et al. [25] also discuss interoperability between CS's in a crisis management SoS. A meta-model is proposed, which is divided into a "treatment system", corresponding to the crisis management SoS, i.e. the system-of-interest, and a "studied system", which is the environment where the crisis occurs. When combining these two ontologies with a reasoning engine, appropriate responses can be deduced.

Zhu et al. [26] propose an OWL2 based ontology for describing SoS missions and discusses principles for how the mission may be decomposed and allocated to CS's. A mission model is analyzed to detect common mistakes prior to execution, and this is done by expressing the mistakes as SPARQL queries and submit them to an OWL reasoning engine. The approach is exemplified through an air defense application. The paper is more detailed in specifying missions than our ontology but is focused on the design of missions rather than SoS operations.

Yahia et al. [27] discuss principles for how ontologies and description logic can be used to achieve interoperability in SoS. 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.

C. Implementation Aspects of Operational SoS Ontologies

Curry [28] proposes the use of linked data for dealing with SoS interoperability, using dataspace architecture that allow for more heterogeneous and distributed storage and querying. 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.

Operationalizing the mission part and combining that with an execution engine can be seen as a way to dynamically adapt autonomous CS's, similarly to the approach in [29].

Durbha et al. [30] discuss interoperability in the Global Earth Observation System of Systems (GEOSS). It discusses possible conflicts between constituent system data, and how to modularize the potentially enormous ontology needed.

Outside the SoS operational context, linked data is also the foundation for the Open Services for Lifecycle Collaboration

(OSLC) standard for interoperability between systems engineering tools across organizations [31].

VIII. Conclusions and Future Research

This paper has investigated, through a case study in road construction, the interoperability of CS's in an SoS based on the use of linked data and ontologies.

The first research question concerned what kinds of information that need to be shared. This was answered by developing a core ontology with concepts such as asset, AAS, organization, capability, mission, and observation.

The second research question was related to the effectiveness of linked data and ontologies as a representation of the required information. A prototype implementation confirmed that this is a fruitful approach, but also provided many insights and experiences regarding how to use the techniques and identified areas where further development is needed.

This research is ongoing and will continue in several directions. Related to the particular case study, we plan to investigate more advanced reasoning techniques and optimization. This includes synthesis of plans, uncertainty assessment including fidelity of data, and preferences. We also plan to extend the work to more domains, with a start in manufacturing and Internet of Things, and to a broader part of the SoS lifecycle. Ultimately, we believe that the solutions discussed in this paper may form the backbone of a DevOps approach for SoS with the world models acting as "digital twins" of the operational SoS.

References

[1] M. W. Maier, "Architecting Principles for Systems-of-Systems," INCOSE Int. Symp., pp. 565–573, Jul. 1996.

[2] T. Berners-Lee, J. Hendler, and O. Lassila, "The Semantic Web," *Scientific American*, vol. 284. pp. 34–43, 2001.

[3] A. R. Hevner, S. T. March, J. Park, and S. Ram, "Design science in information systems research," MIS Q., vol. 28, no. 1, pp. 75–105, 2004.

[4] World Wide Web Consortium. "RDF 1.1 Primer," 2014. URL: https://www.w3.org/TR/rdf11-primer/.

[5] World Wide Web Consortium. "SPARQL 1.1 Overview," 2014. URL: https://www.w3.org/TR/sparql11-overview/.

[6] World Wide Web Consortium. "RDF 1.1 XML Syntax," 2014. URL: https://www.w3.org/TR/rdf-syntax-grammar/.

[7] World Wide Web Consortium. "RDF Turtle 1.1: Terse RDF Triple Language," 2014. URL: https://www.w3.org/TR/turtle/.

[8] World Wide Web Consortium. "RDF Schema 1.1," 2014. URL: https://www.w3.org/TR/rdf-schema/.

[9] World Wide Web Consortium. "OWL 2 Web Ontology Language Quick Reference Guide," 2nd Edition, 2012. URL: https://www.w3.org/TR/owl-quick-reference/.

[10] ISO. "Information and documentation – The Dublin Core metadata element set," Standard No. 15836-1:2017, 2017.

[11] D. Brickley and L. Miller. "FOAF Vocabulary Specification 0.99," 2014. URL: http://xmlns.com/foaf/spec/.

[12] J. Axelsson, J. Fröberg, and P. Eriksson, "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*, 2018, pp. 576–582.

[13] McKinsey & Co. "Reinventing Construction: A Route to Higher Productivity." Feb. 2017.

[14] DIN SPEC 91345. Reference Architecture Model Industrie 4.0 (RAMI4.0). April, 2016.

[15] ISO/IEC. Advanced Message Queueing Protocol (AMQP) v1.0 specifcation. ISO/IEC standard 19464, 2014.

[16] J. Axelsson and A. Kobetski, "Towards a risk analysis method for systems-of-systems based on systems thinking," in IEEE International Systems Conference (SysCon), 2018.

[17] OMG. Business Process Model and Notation (BPMN), version 2.0. 2011.

[18] W. R. Ashby, An Introduction to Cybernetics. London: Chapman & Hall Ltd., 1956.

[19] Oracle. Oracle Spatial and Graph: Benchmarking a Trillion Edges RDF Graph. White paper, Nov. 2016.

[20] E. Sirin, B. Parsia, B. C. Grau, A. Kalyanpur, and Y. Katz, "Pellet: A practical owl-dl reasoner," Web Semantics: science, services and agents on the World Wide Web, Vol. 5, no. 2, 2007.

[21] G. Langford and T. Langford, "The making of a system of systems: Ontology reveals the true nature of emergence," 2017 12th System of Systems Engineering Conference (SoSE), Waikoloa, HI, 2017.

[22] G. Abdalla, C. D. N. Damasceno, M. Guessi, F. Oquendo and E. Y. Nakagawa, "A Systematic Literature Review on Knowledge Representation Approaches for Systems-of-Systems," 2015 IX Brazilian Symposium on Components, Architectures and Reuse Software, Belo Horizonte, 2015, pp. 70-79.

[23] M. B. Sarder and S. Ferreira, "Developing Systems Engineering Ontologies," 2007 IEEE International Conference on System of Systems Engineering, San Antonio, TX, 2007, pp. 1-6.

[24] Y. Baek, J. Song, Y. Shin, S. Park and D. Bae, "A Meta-Model for Representing System-of-Systems Ontologies," 2018 IEEE/ACM 6th International Workshop on Software Engineering for Systems-of-Systems (SESoS), Gothenburg, Sweden, 2018, pp. 1-7.

[25] F. Bénaben, C. Hanachi, M. Lauras, P. Couget, and V. Chapurlat. "A metamodel and its ontology to guide crisis characterization and its collaborative management." Proc. 5th International Conference on Information Systems for Crisis Response and Management, Washington, DC, USA, May, pp. 4-7. 2008.

[26] W. Zhu, H. He and Z. Wang, "Ontology-Based Mission Modeling and Analysis for System of Systems," IEEE International Conference on Internet of Things, Exeter, 2017, pp. 538-544.

[27] E. Yahia, J. Yang, A. Aubry, and H. Panetto, "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. Springer, Berlin, Heidelberg, 2009.

[28] E. Curry, "System of systems information interoperability using a linked dataspace," 2012 7th International Conference on System of Systems Engineering (SoSE), Genova, 2012, pp. 101-106.

[29] J. Axelsson and A. Kobetski, "On the conceptual design of a dynamic component model for reconfigurable AUTOSAR systems," in 5th Workshop on Adaptive and Reconfigurable Embedded Systems, 2013.

[30] S. S. Durbha, R. L. King and N. H. Younan, "An Information Semantics Approach for Knowledge Management and Interoperability for the Global Earth Observation System of Systems," in IEEE Systems Journal, vol. 2, no. 3, pp. 358-365, Sept. 2008.

[31] M. Saadatmand and A. Bucaioni. "OSLC Tool Integration and Systems Engineering--The Relationship between the Two Worlds." 40th EUROMICRO Conference on Software Engineering and Advanced Applications (SEAA). IEEE, 2014.